UNIVERSITY OF CANTERBURY

MASTERS THESIS

Robust Contour Based Surface Reconstruction Algorithms for Applications in Medical Imaging

---

Author:
David MACKAY

Supervisor:
Dr. Ramakrishnan MUKUNDAN

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science

in the

Computer Science and Software Engineering Department

February 24, 2019

UNIVERSITY OF CANTERBURY

Abstract

College of Science

Computer Science and Software Engineering Department

Master of Science

Robust Contour Based Surface Reconstruction Algorithms for Applications in Medical Imaging

by David MACKAY

Contour-based surface reconstruction in medical imaging refers to the reconstruction of a 3D surface from extracted contours from traditional medical imaging modalities such as HRCT or MRI scans. Reconstructed models from HRCT and MRI have a variety of applications including diagnosis, treatment planning, and educational purposes. A notable problem in contour-based surface reconstruction with considerable interest is the branching problem. This problem refers to the issues that arise when attempting to reconstruct branching structures that occur in human anatomy. This research introduces dynamic time warping as a solution to the problem of point correspondence in reconstruction to provide alignments between contours sampled from structures that are known to be problematic. Contour-based surface reconstruction in this application is difficult to test. Often it requires input from a medical practitioner as to whether a reconstruction is accurate or not. However, early testing can be done during development with synthetically generated models. This research provides quantitative analysis with the introduction of a technique for generation of test contour data from ground truth surfaces. The ground truth meshes used in this work represent commonly encountered cases in surface reconstruction. Results from comparative analysis show robustness in the proposed technique with variable levels of source contour information available.

Acknowledgements

I would like to thank Dr. Ramakrishnan Mukundan for his endless advice and discussions throughout the course of this research.

I also thank the University of Canterbury for providing the means necessary to undertake Masters research.

Most importantly I thank my parents, friends and extended family for providing me with the immense support and encouragement needed to dedicate myself to such a monumental task.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Medical imaging is a large area of research in computer graphics and computer vision. Surface reconstruction is widely used for the creation of 3D models of organs from traditional methods of medical imaging such as High-Resolution Computed Tomography (HRCT) and Magnetic Resonance Imaging (MRI). Currently there exist several surface reconstruction tools specifically for applications in medical imaging [1], [2]. Software such as Meshlab [3] provides tools for research in the more general area of surface reconstruction and mesh analysis. Applications of 3D models obtained by reconstructing data from HRCT and MRI include diagnosis, treatment planning, intraoperative support, documentation and educational purposes [4]. Direct volume rendering is a technique that renders a 3D model from medical images without intermediate representation [5]. The Marching Cubes algorithm [6] and its improvements [7] are commonly used as part of direct volume rendering. Indirect surface rendering, also known as indirect volume rendering, is a group of techniques that generate intermediate 3D model representations from HRCT and MRI scans, before rendering of the model. Generation of an intermediate format expands the potential applications of surface reconstruction. An example of such an application is the use of 3D printed models for educational purposes. Although these reconstruction techniques allow for the production of an intermediate format, many of them lead to inaccuracies in reconstructed surfaces. A large number of techniques require manual input or are also partially automatic, meaning they only use optional user input to correct or assist in the segmentation stage of reconstruction. Attempts at fully automatic reconstruction have also been attempted to remove the need for manual user input from this process [8].

Direct volume rendering currently has the advantage that explicit segmentation is not required. Surface rendering requires this step which can be erroneous. However, surface rendering still has advantages such as being able to perform collision detection, fast rendering, simulating mechanical behaviour and applications like 3D printing [9].

1.1 Research Motivation

Contour-based surface reconstruction, also known as indirect surface rendering, is the problem of reconstructing a 3D surface from a series of 2D contours. Such contours can be obtained, for example, by pre-processing axial scan images in an HRCT stack to obtain bounding edges of imaged regions. As there is an infinite number of solutions to a surface reconstruction problem in general [10], [11], constraints must be imposed to obtain meshes that represent the original surface accurately. In medical imaging, surface reconstruction has generated interest for indirect volume rendering, where an intermediate representation is generated from medical images. Issues in reconstruction often occur in indirect volume rendering, especially in the branching case.

Improvements to both accuracy and implementation performance will encourage more widespread adoption of indirect volume rendering in medical imaging. The goal of this thesis is to investigate areas of improvement to indirect volume rendering techniques in medical imaging. Contributions to the understanding of the properties of various surface reconstruction techniques will allow for more informed decisions about algorithm use in applications of medical imaging.

1.2 Aim and Approach

The main aim of this thesis is to investigate the use of Dynamic Time Warping (DTW) in surface reconstruction for applications in medical imaging. Use of this technique is proposed for improvements to reconstruction accuracy for applications in medical imaging, as well as allowing for triangulation in branching cases. The approach for this investigation follows the general process:

- Develop improvements to surface reconstruction accuracy with the introduction of the application of DTW for point correspondence in contour-based surface reconstruction in medical imaging.
- With DTW as the foundation, create a full, automatic contour-based surface reconstruction technique for applications in medical imaging.
- Provide comparative analysis with the application of ground truth testing to contour-based surface reconstruction evaluation.
- Analyse efficiency of the implemented techniques used in surface reconstruction, acknowledging where parallelisation can be implemented to improve implementation performance.

This thesis does not aim to provide an in-depth analysis for all methods of indirect surface reconstruction, nor does it compare reconstructed surfaces with surfaces created in direct volume rendering methods. This research also does not aim to perform segmentation of any internal structures. In general, it is assumed that extracted contours are available and this thesis does not cover methods for producing these extracted contours.

1.3 Thesis Outline

The rest of this thesis is presented as follows:
Chapter 2 provides an in-depth background in the research area of surface reconstruction in general and specifically concerning the application of medical imaging and related work.
Chapter 3 presents the implemented contour-based surface reconstruction technique.
Chapter 4 presents the implementation specific details for the test framework used in analysis.
Chapter 5 provides a comparative analysis of surface reconstruction techniques for contour-based reconstruction.
Chapter 6 summarises the work presented and concludes the thesis. Additionally, future work in this area is outlined.

Chapter 2

Background

This chapter reviews research in surface reconstruction, evaluation and its application to medical imaging in general. Section 2.1 provides a basic understanding of medical imaging concepts required for surface reconstruction. Sections 2.2 and 2.3 cover direct volume rendering and indirect volume rendering respectively. Dynamic Time Warping (DTW) is covered in Section 2.4, benchmarking surface reconstruction techniques in Section 2.5 and finally heterogeneous computing and its applications to surface reconstruction in Section 2.6.

2.1 Medical Imaging and Surface Reconstruction

Medical imaging has a wide variety of uses in modern medicine. Some of the main applications of visualisation include diagnosis, treatment planning, image-guided surgery, biological simulation, documentation and educational purposes [4]. To further aid in these areas, new methods for 3D visualisation of medical images are a current area of research.

2.1.1 Medical Imaging Modalities

Medical images are currently obtained by a variety of modalities. X-ray Computed Tomography (CT) produces cross-sectional images [12] from x-rays that pass through an object. The number of X-rays that are absorbed by an object depends on the composition and density, allowing for inferences about the type of object it passed through. High-resolution CT (HRCT) is a specific type of CT scanning which produces higher resolution cross-sectional images.

Magnetic resonance imaging (MRI) is a more recent modality which also produces image slices [12]. This modality uses the fact that nuclei in human tissue react deterministically when exposed to an external magnetic field. MRI is a direct medical application of nuclear magnetic resonance. MRI can be applied to any part of the human body that contains hydrogen without the use of X-rays or ionising radiation. In relation to CT, MRI can obtain greater contrast between different soft tissues. Another advantage in relation to CT is that MRI can be used to acquire images of blood vessels without contrast injection.

Other medical imaging modalities exist, however in reconstruction this thesis considers contours extracted from HRCT or MRI only. These modalities generate representations of anatomical structures using axial, sagittal or coronal scans, and so could be considered for volumetric or geometric reconstructions.

2.1.2 Segmentation

Segmentation in medical imaging refers to the division of an image into segments with similar properties [13]. Ross, Estépar, Díaz, *et al.* [14] provided a method of lung segmentation that required some user interaction. The software segmented lungs into five distinct regions: the left upper and lower lobes, and the right upper, middle and lower lobes. Adaptive Border Marching (ABM) [15] is a method created to perform lung segmentation as a smoothing process of contours. An advantage of this method is that it does not omit regions such as juxta-pleural nodules, whereas threshold-based region filling methods tend to omit this information. Hausdorff distance can be used to evaluate distance information of two contours [9]. Another measure used for evaluating segmentation results is the Dice-coefficient. This is defined as the overlap of two-pixel sets. Kanzaki, Kikkawa, Sakamoto, *et al.* [16] provides a case study where a 3D pulmonary model of a patient was used to assist in a medical procedure.

Although this thesis does not aim to analyse segmentation methods, it is worth covering briefly as segmentation directly influences indirect volume rendering accuracy. Accurate segmentation of anatomical structures present in a stack of images is needed, up to the desired level of detail. Several types of segmentation methods

are commonly used for this purpose: intensity-based segmentation, morphological operators, texture-based segmentation.

2.2 Direct Volume Rendering

Currently, the two main types of rendering used for 3D medical imaging are direct volume rendering and indirect volume rendering [4], also known as surface rendering. Direct volume rendering involves directly rendering the volumetric data set without the generation of an intermediate data set [12].

Balsa Rodríguez, Gobbetti, Iglesias Guitián, *et al.* [5] provide a survey of state-of-the-art compressed GPU based direct volume rendering methods. This survey covers a variety of methods used for compression in direct volume rendering methods and future research in this area. The authors note that while rendering on desktop environments is well researched, direct volume rendering in a distributed or mobile environment is not as thoroughly explored. The paper also highlights existing issues with compression performance of wavelet-based techniques and other recent techniques based on tensor analysis or sparse coding.

Direct volume rendering is an significant technique in medical imaging and is used in a plethora of open source [1], [17] and commercial software [18]. However, this thesis aims to cover specific improvements to, and analysis of, indirect volume surface reconstruction.

2.2.1 Existing Medical Imaging Frameworks

There exist software frameworks that allow for volume visualisation in medical imaging including Voreen, 3DSlicer, and InViWo. Voreen is a ray-casting based volume visualisation framework that aims to be extensible [1]. The user interface of Voreen allows for data flow to be designed by the user of the framework. It is an open source framework written in C++. Another open source framework for medical imaging is 3D Slicer [17].

Interactive Visualisation Workshop (InViWo) is a visualisation framework that is intended for interactive visualisation research and application development [2]. Similar to Voreen it also allows the user to design the data flow networks. It is also

open source and written in C++. InViWo aims to be more extensible than other visualisation frameworks. C++11, OpenGL and OpenCL are supported through extensions. The framework is designed in a modular fashion, allowing for graphics API implementation that is independent of its core functionality.

There is a considerable amount of medical imaging frameworks that provide direct volume rendering functionality; this thesis does not plan to cover all of them. In its current state, surface rendering is not currently implemented and distributed as widely as direct volume rendering. With greater reconstruction accuracy and performance this may change in the future.

2.3 Indirect Volume Rendering (Surface Reconstruction)

Surface rendering is a method used for medical imaging that generates an intermediate format before rendering the image. It is also called indirect volume rendering because of this intermediate representation. One advantage of indirect volume rendering in relation to direct volume rendering is its portability, benefiting recent applications such as 3D printing. Models obtained from medical imaging may be printed for educational purposes with the use of an intermediate format generating in surface rendering. Additionally, these models may also be printed for collaborative discussion about treatment in professional scenarios [4].

Indirect volume rendering relies on various stages to output a reconstructed mesh. This thesis primarily considers those stages of reconstruction that follow after contours have been extracted from input data. However, it is worth noting that improved contour extraction techniques will allow for better overall accuracy in reconstruction in the later stages.

Commonly contour-based surface reconstruction is split into three distinct stages of contour matching, or contour correspondence, and triangulation. Previous work has also added extra stages of reconstruction such as pre-processing techniques, for example, contour interpolation between slices [10].

2.3.1 The Branching Problem

The branching problem is a commonly encountered issue with surface reconstruction from contours [10]. The branching problem refers to the case where contours are encountered that appear to match to contours on a neighbouring slice in such a way that is one-to-many, or many-to-many. This problem introduces a considerable number of cases that must be considered. A goal of this thesis is to solve some of the more commonly encountered issues of branching in medical imaging. One case that is encountered in medical imaging of lung anatomy is bifurcations of the trachea and further subdivisions of bronchi forming the airway tree [19]. These subdivisions result in a tree-like structure which presents many branching cases with each branch resulting in smaller contour sizes. The specific issue that arises with branching is the ambiguity it introduces with contour correspondence.

Figure 2.1 demonstrates how ambiguity can be produced with three contours C_1, C_2, and C_3. C_1 and C_2 are on the same slice. There have four possible configurations:

1. No contours correspond to each other.
2. C_1 is matched to C_3, C_2 is not.
3. C_2 is matched to C_3, C_1 is not.
4. C_1 and C_2 are both matched to C_3. This is a branching condition.

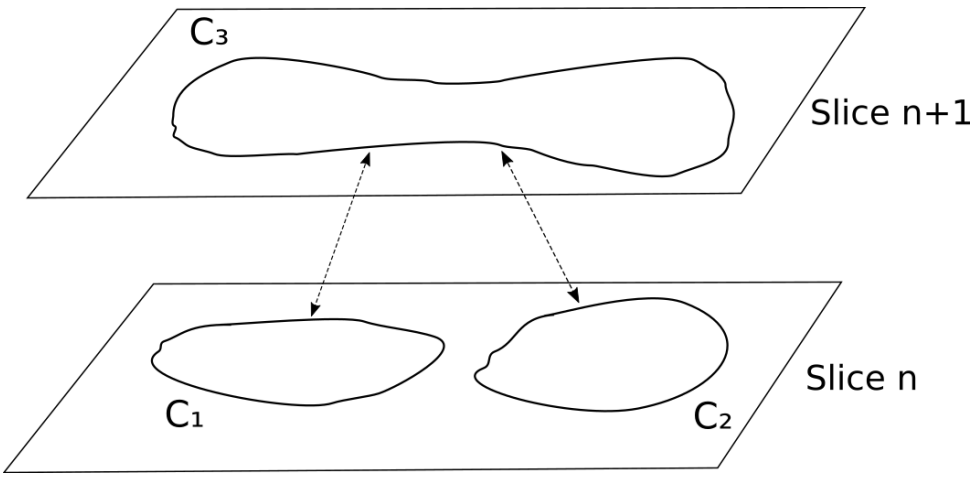


FIGURE 2.1: Ambiguity in branching.

Resolution of branching ambiguity is required in contour-based surface reconstruction, regardless of whether it is an implicit or explicit stage of reconstruction.

2.3.2 Marching Cubes

The marching cubes algorithm is a well-known method for generating a 3D Mesh from voxels [6]. This method uses a look-up table to determine how a surface intersects a cube. Eight vertices are generated from four corners in two slices; this forms the cube. Once a surface has been determined, the algorithm marches to the next cube. Since its creation, there have been improvements to this method such as resolving the problem with ambiguity present in the original [7], [20]. Marching cubes is commonly used for direct volume rendering where the voxels are treated as a 3D lattice [4], [12], [21].

Marching cubes can also be used to compute piecewise parts of a surface from an implicitly defined surface. This method can be used for reconstruction of surfaces from point cloud fitting techniques such as Algebraic Point Set Surfaces (APSS) [22] and Robust Implicit Moving Least Squares (RIMLS) [23]. Open source implementations of these techniques are available in Meshlab [3].

2.3.3 Other Reconstruction Techniques

Voronoi Filtering is used for surface reconstruction in work by Amenta and Bern [24]. This work extends upon previous research that reconstructed curves from sample points in 2D [25]. This method works in 3D and generates a Voronoi Diagram from sample points. This algorithm works well in general for reconstruction when there is a large number of sample points. It also only requires sample points, and it does not need normal information. However, there is a noted issue with reconstruction when there is under-sampling. This type of surface reconstruction is also conveniently available in MeshLab [3] and it appears to work well in practical situations.

Barequet, Shapiro, and Tal [26] presents an algorithm with generates a triangulation from a series of cross sections. The method consists of two stages. The first stage implements contour matching and tiling, which is to decide which contours match between cross sections automatically. The second stage involves generating a triangulation based on the matching of contours in the first stage. The reconstructed

surface can then be used for the 3D representation of the original object. As a metric for accuracy, Hausdorff distance on created test models was used. This work is extended upon by Barequet, Goodrich, Levi-Steiner, *et al.* [27] where a method of contour interpolation by straight skeletons is introduced. The authors present a technique for interpolating a piece-wise linear surface between slices. This work is again extended upon by Barequet and Vaxman [10] where they improve the existing method by introducing non-linear interpolation between slices. This method aims to visualise a 3D object given only by its level sets. The authors explain how the definition of a heuristic is required in surface reconstruction as the problem itself has an infinite number of solutions. One issue explored by the authors was correspondence which involves other problems like branching and geometric dissimilarities.

Li, Belkasim, Pan, *et al.* [28] defines a method of 3D reconstruction based on mapping contours between images. The authors define a tree-like contour data structure to store object model information efficiently.

Mukundan [8] proposes a method for automatic construction of 3D meshes from HRCT scans. The method identifies regions of interest, extracts contours before performing point matching to generate the mesh between slices. Points between contours were matched based on minimum Euclidean distance. This method was simple but was noted to encounter issues with branching. The advantage of this method is that it produces a portable high polygon model. It also noted that the solution was implemented in a way that could be parallelised. A proof of concept of a parallel implementation was created to show the benefits of parallelisation in slice-based reconstruction [29].

A method of 3D reconstruction of a human bronchial tree is presented by Fetita and Preteux [30]. Mathematical morphology is used for 2D segmentation of slices and a propagation technique is used for the 3D reconstruction.

Shin and Jung [31] introduces a method for reconstruction of surfaces based on 2D contour lines. Contour matching is performed by projecting vertices onto adjacent slices. If the projected vertices are within a contour on an adjacent slice, then the contours are matched. Minimum Euclidean distance is then used to match vertices between contours.

Kienel, Vančo, Brunnett, *et al.* [32] uses active contour detection to provide semi-automatic detection of embryonic contours. Partial manual input was used to prevent the over and under segmenting that automatic segmentation is susceptible to. For surface reconstruction, contours are matched based on whether convex hulls of contours overlap. Triangle edges are generated based on Euclidean distance and tangent angle.

Sopabutra, Horkaewa, and Gansawat [33] introduces a method for geometric reconstruction for ultrasonography. The authors also provide a method of benchmarking geometric reconstruction accuracy. Both synthetic and real models are used for testing the accuracy and performance of the surface reconstruction. A method of 3D airway reconstruction based on centre-lines is provided by Miyawaki, Tawhai, Hoffman, *et al.* [34]. This technique constructs a geometric structure from a 1D tree. An advantage of this type of reconstruction is that the 1D tree gives additional information about branching such as connectivity, coordinates of branching points and generation numbers. Applications such as computational fluid dynamics can take advantage of this additional information. The method can construct bifurcations similar to previous work; additionally, it can reconstruct trifurcations. It is not able to perform reconstruction for branching involving more than three child branches, however.

Akkouche and Galin [35] introduces a method for reconstructing implicit surfaces from contours. The paper aims to provide a solution towards correspondence and branching problems that arise in 3D reconstruction for medical imaging.

2.3.4 Point Cloud Surface Reconstruction

A subset of surface rendering techniques that has a considerable amount of research towards it is point cloud surface rendering. This research field is closely related to surface reconstruction in medical imaging, with some attempts to use point cloud reconstruction techniques in medical imaging.

A recent survey paper on point-cloud surface reconstruction methods covers a variety of techniques [11]. It is explained how an infinite number of surfaces can pass through, or near, points in a point cloud. Because of this certain assumptions and constraints have to be made about the reconstructed surface. The survey paper

covers common assumptions shared by techniques and compares the relative advantages and disadvantages of these assumptions. These advantages lead to greater knowledge about how techniques are better suited for certain applications. One set of assumptions is that about surface smoothness. The paper covers different ways that methods take advantage of how smooth a surface should be. This smoothness may be local, global, or piece-wise. Additionally, the paper covers some common evaluation techniques for surface reconstruction. Some of these techniques include Hausdorff distance, mean distance to points and normal errors.

Manson, Petrova, and Schaefer [36] investigates the use of wavelets for surface reconstruction. The authors show that the wavelet-based surface reconstruction can be used in combination with octrees as an efficient method of smooth surface reconstruction. Octrees are used to store the wavelet basis in a hierarchical fashion.

Moriconi, Scalco, Broggi, *et al.* [37] extends and uses the method proposed by Manson, Petrova, and Schaefer [36] specifically for 3D surface reconstruction of medical images. The method also employs a pre-processing step that interpolates binary segmentations to generate a new slice between two existing ones. This process is called the inter-slice thickening step. Similar to Manson, Petrova, and Schaefer [36], this method implements octrees as a form of hierarchical representation of the point cloud. The maximum depth of the octree can be limited to improve the performance of the algorithm. A higher depth octree configuration will lead to a smoother surface. Another method which extracts point sets from contours for implicit surface reconstruction is proposed by Braude, Marker, Museth, *et al.* [38]. This method uses a multi-level partition of unity implicit models to create an approximate fit. Surface normal estimation is done by creating a binary volume and calculating the gradient of the blurred volume. A downside to this normal estimation method is that although it uses extra information from the contours, it may not work well for estimating normal information for internal structure. Approximation error is calculated using medical imaging data sets.

Smooth Signed Distance Surface Reconstruction [39] takes an oriented point cloud as input and outputs a watertight surface. The method forces the implicit function to smoothly approximate the signed distance to the surface. This method shows greater performance when the points are irregularly sampled.

Kazhdan and Hoppe [40] also introduces a surface reconstruction method based on oriented point clouds. This method uses a modified version of Poisson reconstruction to reconstruct a surface. The original Poisson reconstruction method tended to over-smooth the data and so important features would be missed. The modified screened Poisson surface reconstruction [40] introduced constraints so that the over-smoothed features would now be included in the final reconstruction. This however introduced some problems when there was a large amount of noise. The paper also compares the method with the original Poisson reconstruction, wavelet-based surface reconstruction [36] and smooth signed distance reconstruction [39]. The paper concludes that in general screened Poisson surface reconstruction provided improved geometric quality when compared to the original Poisson surface reconstruction method. It also provided comparable quality to the smooth signed distance reconstruction method, as well as being a much more efficient method. The paper also outlines several areas of improvement where the method may be improved in future. OpenMP was used to provided multi-threaded processing, but GPU parallelisation was suggested as future work.

Digne, Cohen-Steiner, Alliez, *et al.* [41] introduces a method for surface reconstruction which uses the optimal transport problem to adjust a surface constructed by Delaunay Triangulation. This solution currently has issues with performance as the optimal transport problem is a linear programming problem.

Xiong, Zhang, Zheng, *et al.* [42] presents a method of surface reconstruction from point clouds from dictionary learning. This method did not require point normal information but could use this if provided. The method is comparable in accuracy to other point cloud methods but can generate holes in the reconstructed surface.

When point cloud surface reconstruction is applied to contours, the contours are treated as part of the point cloud. Treating contour information as points this manner omits certain information about the contours which can affect reconstruction accuracy. Techniques that reconstruct implicit surfaces have the advantage of generating "watertight" surfaces [36], [39], [40]. However, a disadvantage of these implicit surface reconstruction techniques is that the approximating surfaces are not guaranteed to pass through source points.

Point Cloud Normal Estimation

Many surface reconstruction techniques require normal information or estimations to function correctly. One such method of normal estimation uses least squares to determine an estimate of point normal information given a point cloud as input [43]. The sum of square distances from points to a plane is minimised for a neighbourhood of points. This is done for each point in the point cloud to get a normal estimation for all points in the point cloud. Demarsin, Vanderstraeten, Volodine, *et al.* [44] estimates point normal information as the normal of least squares plane to the 1 ring neighbourhood of the point. Li, Schnabel, Klein, *et al.* [45] provides a method of point cloud estimation which improves on existing methods with noise removal and sharp feature detection.

2.3.5 Pre-processing Techniques

Dynamic elastic interpolation is a method that was proposed for contour interpolation in medical imaging [46]. This method shows the advantage of contour interpolation for handling the branching problem.

A method of interpolating between contours is also presented by Barrett, Mortensen, and Taylor [47]. The method does not require storing of intermediate data structures when processing. The paper shows its usefulness for interpolating terrain contours as well as applications in medical imaging. The method interpolates between contours which may involve branching. Mathematical morphology is used to perform the interpolation between contours. The authors also state that the method can be parallelised. A similar method is proposed by Guo, Cai, and Wang [48] which applies morphological operations to contour interpolation for 3D reconstruction in medical imaging. A method of interpolation between slices is provided by Frakes, Conrad, Healy, *et al.* [49]. This method uses control grid interpolation for inter-slice reconstruction. The inter-slice information is used for vascular morphological surface reconstruction. A method of generating intermediate contours using level set analysis is introduced by Mukherjee and Ray [50]. The method uses shape specific features and optical flow vectors to generate interpolated contours. The authors

show applications for time series data such as path tracing cyclones for meteorological applications as well as spatial series data such as 3D organ visualisation for medical imaging.

2.4 Shape Matching and Dynamic Time Warping

DTW is a widely used dynamic programming technique that is suitable for analysis of temporal sequences [51]. Given two series of observations $X = \{x_0, ..., x_n\}$ and $Y = \{y_0, ..., y_m\}$ DTW obtains an optimal alignment between these series under a specific set of constraints. A $n \times m$ matrix can be constructed containing the cumulative cost of alignment for all possible alignments. The cost is defined by some objective function. With this matrix a path can be extracted that minimises the warping cost.

2.4.1 Recurrence Relation

Silva and Batista [52] define initial conditions and a recurrence relation for creating DTW matrices. i and j are defined as indices for feature sequences X and Y respectively.

The initial condition is defined in Equation 2.1.

$$dtw(i,j) = \begin{cases} \infty, & \text{if } i = 0 \text{ or } j = 0. \\ 0, & \text{if } i = j = 0. \end{cases} \tag{2.1}$$

The recurrence relation is defined in Equation 2.2. Note that $c(x_i, y_i)$ is the cost function for observations x_i and y_i.

$$dtw(i,w) = c(i,j) + min \begin{cases} dtw(i-1,j) \\ dtw(i,j-1) \\ dtw(i-1,j-1) \end{cases} \tag{2.2}$$

The way this recurrence relation affects point correspondence in shape matching is shown in Figure 2.2.

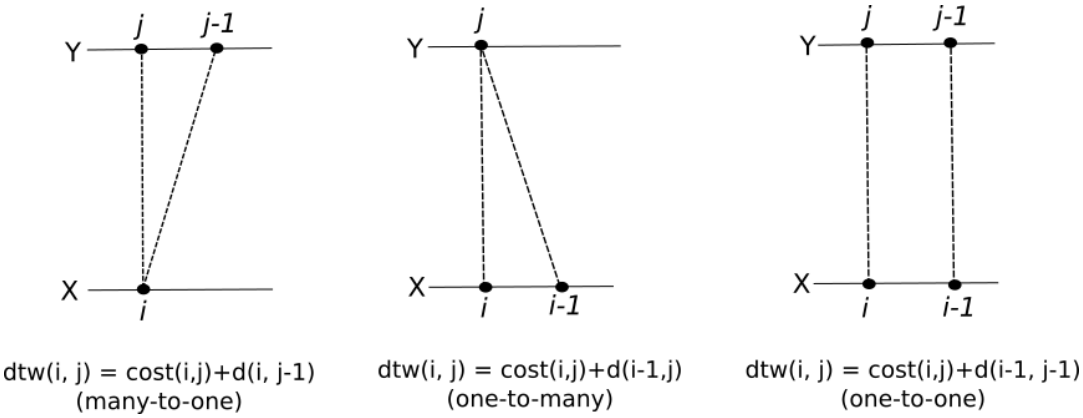


FIGURE 2.2: Dynamic time warping in point matching.

2.4.2 Warping Path Constraints

DTW has certain restrictions on the warping path which were developed as part of its original application in speech recognition [53]. These restrictions include the monotonic, continuity and boundary constraints. These are required for generation of the DTW matrix. Global constraints such as the Sakoe-Chiba band and slope constraint are also used. The constraints are defined as follows:

Monotonic conditions:

$$i_{k-1} \leq i_k$$

$$j_{k-1} \leq j_k$$

This means that the indices i and j of the warping path will not decrease. They can only increase or stay the same.

Continuity conditions:

$$i_k - i_{k-1} \leq 1$$

$$j_k - j_{k-1} \leq 1$$

The continuity condition restricts warping path index increases to no more than one.

Boundary conditions:

$$i_1 = 1, j_1 = 1$$

$$i_K = I, j_K = J$$

Where I is the maximum index of sequence X and J is the maximum index of sequence Y.

This condition restricts the boundary of the warping path. The minimum i index value and j value are assumed to match to each other. Similarly, the maximum index values are assumed to be part of the warping path.

Global Constraints:

Global constraints have been introduced into DTW. These are used to restrict the warping path for specific applications. One commonly used global constraint is the Sakoe-Chiba band [53], defined in Equation 2.3.

$$|i(k) - j(k)| \leq r \tag{2.3}$$

In the Sakoe-Chiba band, r is a constant value defined to prevent the warping path from straying too far from the diagonal of the DTW table. This restriction is commonly referred to as the "warping window" and is often used as an optional constraint depending on the application. For specific applications, this condition can result in greater accuracy in alignment. Another common global constraint is the Itakura parallelogram [54].

Slope constraint condition:

The slope constraint is another locality constraint which prevents "steep" or "shallow" gradients in the warping path. It is also another constraint which is not always used depending on the application. The slope constraint will help to prevent a short section of a series being matched to a considerably longer section of another series.

The boundary, monotonic and continuity conditions are the main components of dynamic time warping. The adjustment window and slope constraint conditions are not always used as they restrict the deviation from the diagonal of the warping path and avoid steep gradients. These two locality constraints were initially applicable in speech recognition. However, other research has since omitted their use.

2.4.3 Applications and Performance

The complexity of classical DTW is quadratic and is proportional to the length of feature sequences [51]. Prätzlich, Driedger, and Müller [55] identifies work made towards improving the efficiency of dynamic time warping and aims to further develop on this. A method is devised which takes ideas from two existing techniques to create a new one. The method combines both memory restricted DTW and multi-scale DTW to reduce the complexity. The method is successful in reducing the amount of space and computational requirements for DTW. However, it is noted that depending on certain parameters, a globally optimal warping path may not be found. The paper also claims that the method can be parallelised. DTW has also found applications such as shape matching where the data is non-time series [56].

Work has also been done by Silva and Batista [52] to improve the speed of DTW. The authors compare this method to FastDTW as it shares similarities to their work. A difference in the PrunedDTW algorithm proposed by the authors is that it does not rely on best-so-far distance approximations. It is shown to have similar performance improvements to FastDTW without relying on distance approximations. In relation to traditional DTW, PrunedDTW is shown to improve by 50% on average. However, when warping window restrictions are introduced the authors found little improvement in using PrunedDTW over classical DTW.

The importance of Shape Matching and Dynamic Time Warping in this thesis is that the point matching stage of surface reconstruction is treated as a shape matching problem. Dynamic Time Warping is a promising technique for this as it has previously shown success in shape matching [56], [57].

2.5 Surface Reconstruction Benchmarking

Testing accuracy for surface reconstruction introduces another area of research. In the past, some research in surface reconstruction has used direct visual comparison. An issue with this is that it is considered to be subjective, and it is easier to miss finer details such as self-intersections and thin holes. Surface simplification and surface

reconstruction (for both point clouds and contour-based reconstruction) are significant areas of research that require similar forms of mesh accuracy verification and evaluation.

Metro [58] is a command line tool developed for sampling points and measuring error values between surfaces using Hausdorff distance. The authors write about applications in surface simplification accuracy evaluation, and this work is also extended in MeshLab [3]. Polygon Mesh Comparison tool (PolyMeCo) [59] extends on Metro and aims to provide a mesh comparison tool that is user-friendly and has several visualisation options. It extends on other previous work in mesh comparison. MESH [60] is another tool that uses Hausdorff distance for measuring errors between surfaces. Similar to other tools, it only aims to compute these distances for triangular meshes.

Bellmann, Hellwich, Rodehorst, *et al.* [61] introduces a benchmark which defines metrics for image-based surface reconstruction. A ground truth is generated from a laser scanner. This ground truth contains smooth surfaces as well as piece-wise planar patches to provide a range of models for testing.

Berger, Levine, Nonato, *et al.* [62] provides benchmarking techniques for surface reconstruction methods that use point clouds. Synthetic models are created for testing the accuracy of the surface reconstructions. The models used have differing amounts of surface smoothness and sharp features. These features allow for a good range of model shapes for benchmarking and evaluation of these methods. The range of model variability showed which methods were more susceptible to noise and which modelled sharper features better. Point clouds were generated by doing range scans on the test models similar to a real-world scan. The authors found that some error metrics were unsuitable, and so care was taken to construct metrics that showed inaccuracies well.

MeshLab [3] is a free open source mesh processing tool that is designed with research use in mind. Being free and open source software encourages reproducible research results. A variety of input/output formats are supported, and the software is extensible with the support of plugins. It has considerable popularity in surface reconstruction research. As well as implementing surface reconstruction techniques such as screened Poisson surface reconstruction [40], MeshLab also includes tools

for computing vertex quality, Hausdorff distance computation, and blue noise sampling.

A comparison of surface reconstruction techniques specifically for robotic applications is given by Wiemann, Annuth, Lingemann, *et al.* [63]. This paper considers freely available open source implementations of surface reconstruction techniques.

2.5.1 Hausdorff Distance

Hausdorff distance has been used for evaluating accuracy in both surface simplification algorithms and surface reconstruction algorithms [62], [64]. It is worth noting that this metric is not symmetric [65]. The distance from a sample point a_i on mesh A to the closest point b_i on mesh B may not be the same as the distance from the sample point b_i on mesh B to the closest point on mesh A. This is where the asymmetry in Hausdorff distance measurements lie.

A definition of Hausdorff distance is given by Huttenlocher, Klanderman, and Rucklidge [65]. The definition is summarised as follows:
Given two finite point sets $A = \{a_1,...,a_p\}$ and $B = \{b_1,...,b_q\}$, asymmetric Hausdorff distance is given by:

$$h(A,B) = \max_{a \in A b \in B} ||a - b|| \tag{2.4}$$

The symmetric Hausdorff distance is then given by:

$$H(A,B) = \max(h(A,B), h(B,A)) \tag{2.5}$$

2.5.2 Point Sampling

For sampling points on a surface for test data in point cloud surface reconstruction, blue noise sampling is suitable. Blue noise sampling refers to the production of a random uniform distribution of points. One such method of doing this is Poisson disk sampling [66]. A parallel implementation is provided by Bowers, Wang, Wei, *et al.* [67]. This type of sampling is advantageous because it guarantees random sample

points are a certain distance from each other. When considering contours sampled from images in medical imaging; however, the points in real data are sampled uniformly from pixels. For this reason, randomness in sampling does not need to be considered when generating test data for contours.

Ground Truth Testing

In surface simplification, the original mesh is available for direct comparison with a simplified one. However, the same often cannot be done with surface reconstruction. In both point-cloud surface reconstruction and contour-based surface reconstruction, the original surfaces are unavailable. This means that metrics like Hausdorff distance are unusable as there is nothing to compare with. This lack of ground truth prompted researchers in surface reconstruction to create or use existing meshes as test meshes [26], [62]. Test data sets are often created by sampling points on these meshes. Researchers can then use Hausdorff distance to compare with the original test meshes.

Often test meshes are carefully chosen for specific features. For example, in a recent comparison on point cloud surface reconstruction [62], the authors use meshes which have multiple levels of detail as well as a combination of sharp and smooth features. Interestingly they also make use of implicit surfaces for point cloud sampling as test data. They uniformly re-sample the points again in a dense manner and then use this for positional and normal error metrics. Implicit surfaces are used in this benchmarking tool as they better represent smooth surfaces when compared to piecewise planar surfaces. The implicit surface representation is a simple extension upon the multi-level partition of unity implicits [68]. This research generates implicit surfaces from point sets. In the benchmarking tool mentioned [62] this is used to generate implicit surfaces from test models.

2.5.3 Test Model Generation

Although the following techniques have not been used for ground truth testing, this thesis considers the potential for applications in this area.

Tree model generation is an area that is well covered. An earlier research endeavour for this task was done by Lluch, Vivó, and Monserrat [69]. This method provides a way of generating tree structures based on L-systems.

An algorithm was proposed by Kitaoka, Takaki, and Suki [70] for generating models of lung airways. The algorithm proposed makes use of a set of rules which are defined by a set of observations and facts about how bronchial trees are shaped. The authors use these rules to generate bronchial tree models which are close to real models. Pluta, Janaszewski, and Postolski [19] extends this algorithm by introducing some modifications that attempt to make the generated model more realistic. The extensions that the authors added include bending of branches and geometric deformations. A more recent method of generating branching structures is proposed by Mattingly, Chariker, Paris, *et al.* [71]. The authors aimed to generate branching structures for modelling anatomical structures. The algorithm takes a skeleton structure represented by connected node positions and sizes. Two significant vertex generation steps are outlined. Boundary vertices for nodes are generated in one algorithm. Convex hull vertex generation is performed in another.

2.6 Heterogenous Computing in Surface Reconstruction

Open Computing Language (OpenCL) is an open standard for Heterogeneous parallel computing [72]. An extensive introduction to heterogeneous computing with OpenCL is provided in [73]. Implementing heterogeneous computing allows for specific hardware to be combined for a specific use case. This makes it suitable for applications such as use in mobile devices or distributed computing. The three most common forms of parallelism are instruction parallelism, data parallelism, and thread parallelism.

A competing platform for parallel computing is Nvidia's CUDA [74]. An introduction to parallel computing with CUDA is provided by Cook [75]. An in-depth survey on heterogeneous computing techniques is covered by Mittal and Vetter [76]. Additionally, factors and challenges in heterogeneous computing are covered as well as benchmarking methods. A comparison between different frameworks is also covered. A performance comparison between OpenCL and CUDA is introduced

by Karimi, Dickson, and Hamze [77]. In this comparison kernels with minimal difference are written in OpenCL and CUDA. The performance of these two APIs is then directly compared using an Nvidia GPU. It is found that in general, the performance of CUDA was slightly better than OpenCL. However, this has likely become outdated due to changing hardware and specifications. The authors also state that the choice between the two APIs is largely dependent on other factors such as programmer familiarity or whether the development tools are available for the target GPU.

A justification as to why OpenCL tends to perform worse in real-world applications than CUDA is provided in another study [78]. It is noted that OpenCL's portability has made it the subject of criticism that it does not perform as well as CUDA for similar applications. The authors show that the differences in performance are largely due to "unfair comparisons". A fairer performance comparison is provided which shows that OpenCL and CUDA both perform very similarly under these circumstances. The reasons for the gaps in performance in "unfair" comparisons are investigated and attributed mostly to behaviours of programmers, compilers, and users.

Memeti, Li, Pllana, *et al.* [79] provide a more recent comparison of heterogeneous and parallel programming platforms. OpenCL, OpenACC, OpenMP, and CUDA are all compared in terms of performance, power usage, and programming productivity. Productivity was measured with a tool created by the authors. This tool measured the amount of framework-specific lines of code required to develop parallel code. It was found that OpenCL required more programming effort than OpenACC and CUDA. The performance and power consumption differences between OpenCL and CUDA was shown to be small in most cases. For large scale applications, this may mean that it would be suitable to investigate which framework is more efficient on specific hardware and software. A limitation to CUDA and OpenACC is that they only officially support specific devices such as Nvidia GPUs. OpenCL, however, aims to be able to be used on both the CPU and GPU of a system.

Alsmirat, Jararweh, Al-Ayyoub, *et al.* [80] investigated the use of parallel computing in medical imaging. A method was proposed for performing fuzzy clustering for image segmentation in parallel. The author claims that the use of fuzzy c-means

segmentation was suitable because of the segmentation accuracy it provides. The paper outlines four distinct parts of the clustering algorithm that are ideal for parallelisation. Two of these parts run on the CPU and two are performed on the GPU. The paper also notes overhead issues when sending data to the GPU.

Agulleiro, Vazquez, Garzon, *et al.* [81] presents a method of tomographic reconstruction that uses a hybrid of CPU and GPU processing. They find that multi-core computers perform similar, or better, to pure CPU or GPU solutions when multi-threading and vector processing is used.

The research done by Suppan [29] shows that the 3D mesh construction method proposed by Mukundan [8] can be parallelised. The method here used a simpler method for generation of triangles in order to allow for easy parallelisation with OpenCL. This research showed that there are substantial performance gains to be made by implementing GPU based parallelisation. Because this work was intended as a proof of concept, the final method used did not have the same accuracy or complexity as the original method.

Bolitho, Kazhdan, Burns, *et al.* [82] shows that the method of Poisson surface reconstruction can be performed in parallel with distributed memory. The paper successfully shows equivalence between the serial method and parallel method by comparing error for reconstructed surfaces from the same point set. This method is shown to have considerable speed and scalability improvements.

Overall literature shows that surface reconstruction is a highly parallelisable problem. More specifically, in contour-based reconstruction, pairs of consecutive slices can be processed in parallel for reconstruction of surfaces between slices. Multiple areas of this problem can be identified that can be implemented in parallel, allowing for a greater choice of granularity.

2.7 Summary

Literature acknowledges that the state of indirect volume rendering for applications in medical imaging requires further research into techniques with greater accuracy and performance. Contour-based reconstruction techniques have historically had issues with the branching problem. Correspondence of branching contours in medical

imaging proves to be a challenging task. Currently, there exists an acknowledged lack of accuracy in specifically in the reconstruction of branching structures. Both contour-based surface reconstruction and point-cloud surface reconstruction techniques have been applied to surface reconstruction in medical imaging. Decisions for which technique is most appropriate for applications in medical imaging is unclear due to a lack of in-depth comparative analysis. What has been acknowledged by the literature is that there is room for improvement in accuracy.

Comparative analysis has previously been done for the more general problem of point-cloud surface reconstruction, and specifically for applications such as robotics. Further analysis is needed to provide algorithm implementers with the knowledge required to choose the right method for applications in medical imaging. Designing a test framework for applying surface reconstruction to medical imaging, similar to those used in robotics and point-cloud surface reconstruction evaluation, will allow for greater comparative analysis specifically in this area.

Chapter 3

Materials and Methods in Surface Reconstruction

This chapter covers the proposed reconstruction technique. Section 3.1 summarises of the motivating data set and the works that this research builds upon. The following Section 3.2 defines the problem of contour-based surface reconstruction. Sections 3.3 and 3.4 detail how the distinct stages of contour correspondence and point correspondence are implemented in the proposed method respectively. Section 3.4 also covers how dynamic time warping is applied to the problem of point correspondence in surface reconstruction, as well as the reasoning behind the use of this technique. This is followed by Section 3.5 which explains how triangulation is performed using the warping path of the dynamic time warping algorithm. A solution to problems which may arise when triangulating branching cases is provided in Section 3.6. Finally, post-processing techniques are briefly mentioned in Section 3.7.

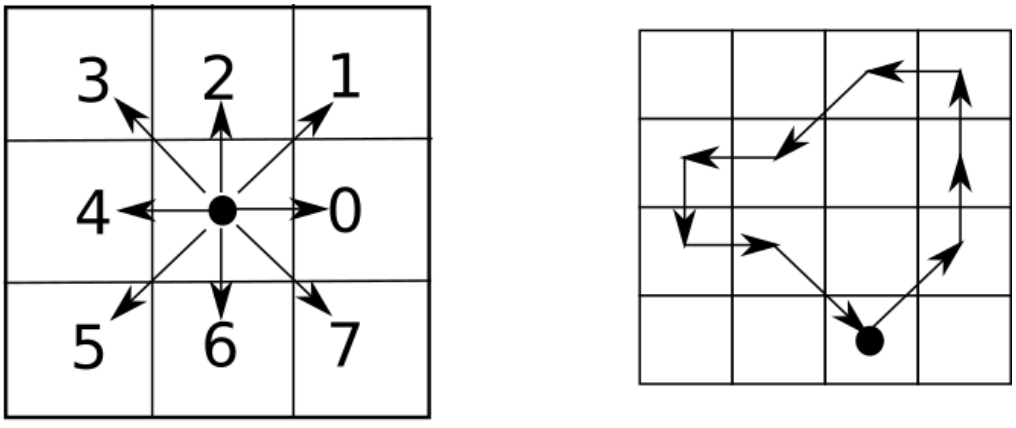
3.1 Materials in Contour-Based Surface Reconstruction

One case study for surface reconstruction in medical imaging is done by Mukundan [8]. This thesis focuses on issues that arise in this real data set. This research uses anonymised HRCT stacks supplied by Dr. Anthony Butler and his Bio-engineering Group at the University of Otago, Christchurch. The use of these HRCT images for this research was approved by the Health and Disability Ethics Committee, Ministry of Health (No. URB/10/02/EXP). Contours are extracted from HRCT stacks of lungs. The data consists of 210 images which are 512x512 pixels. The inter-slice

distance of these images is a constant value of 1.25mm. This particular set of images represents complex airways with multiple branching structures. The majority of contours extracted from this data set contain less than 200 points. It is also observed that a large number of contours are less than size 20 [8], significantly contributing to the complexity of contour correspondence.

3.1.1 Contour data structure

An example of how extracted real data can be represented is given by Mukundan [8]. Points within contours are spaced apart at a fixed distance, only the direction of the next point is stored. This technique minimises the storage overhead of contours. The representation given within assumes that the contour is defined using a set of consecutive (connected) pixels, forming a closed polygonal line. The first point in the contour is stored as x and y values. The remaining points are stored as a relative direction from the previous point. This direction is stored as a single byte which contains a single value from 0 to 7. Since the distance between contours is known, and the distance between points is known, the 3D point representations of contours can be inferred during reconstruction.



a) Point number and direction from current point b) Example point traversal

FIGURE 3.1: Contour directional offset representation.

Figure 3.1a shows how the directional offset points to the next pixel. Figure 3.1b provides an example of a traversal. If this traversal starts at the marked point, then the sequence followed is 1, 2, 2, 4, 5, 4, 6, 0, 7. Given the starting position, the byte value can be used to traverse all points in a contour.

3.2 Surface Reconstruction

The goal of the surface reconstruction algorithm is to take contours extracted from HRCT or MRI slices and generate a surface that represents the real organ as accurately as possible. The required accuracy and performance of the reconstruction technique generally depend on the application. For example, a 3D printed model used for education may not require the same accuracy as a surface used for diagnosis in medical imaging. Generating a mesh from a set of points is a problem that has infinite solutions [10], [11]. To reconstruct a unique solution to the reconstruction problem from a set of contours additional constraints must be established.

In contour-based reconstruction, there is additional information along with point set data. Contours are defined in the context of surface reconstruction in medical imaging as ordered point sets. More specifically a cyclic order can be defined on the set of points in a contour. This ordering of points is shown in Figure 3.2. Contours are also expected to be non-intersecting, that is, contours are not expected to intersect with itself or other contours within the same slice.

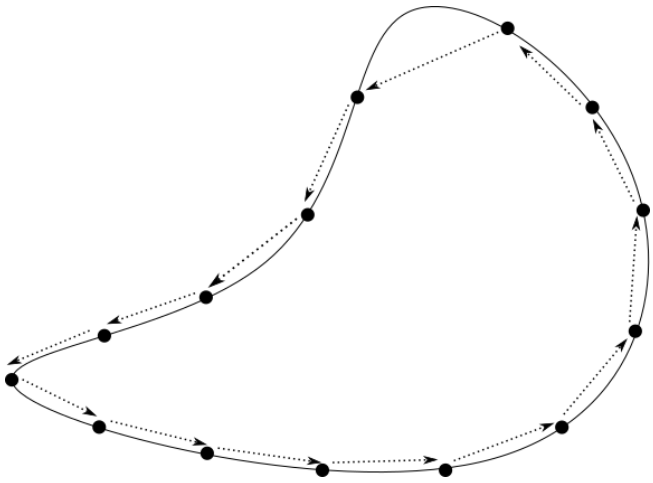


FIGURE 3.2: The ordering of points in a contour.

A slice is defined in this context as a set of contours on a plane perpendicular to a fixed axis. The data is stored in layers, each containing one or more slices. Treating data as a set of unordered points reduces contour-based reconstruction to the more general problem of point-cloud reconstruction, but this omits information about the extracted contours by doing so. Instead, the proposed reconstruction technique aims to use as much information about source contours as possible.

With this information, the problem of contour-based surface reconstruction can be posed as follows. Given a set of n slices $\{S_1, ..., S_n\}$, where a slice S_i can contain zero or more contours on a fixed axis, construct a surface M such that every point from all contours is approximated or included in the final mesh. Because this is vague the following additional goals are introduced:

- The number of holes in the mesh should be minimised.
- Triangle self-intersections should be minimised.
- The technique should be general enough that it requires minimal user input.
- Surface normal information should be correctly established. There should be no back-facing triangles.

Some of these reconstruction goals are met by point-cloud surface reconstruction techniques. The resulting reconstructed mesh from Screened Poisson surface reconstruction [40] for example is two-manifold. This property means that the generated mesh will have no triangle-self intersections. However, this particular technique also approximates the points in surface reconstruction, instead of using sample points directly as vertices in the produced mesh. This property is beneficial in point-cloud surface reconstruction as sample data can contain noise. The data sets extracted for contour-based reconstruction, however, have noise removed in the contour extraction stage. For this reason, the proposed method assumes that all points in contours should be included in the final mesh.

Indirect surface reconstruction algorithms that utilise contour information generally follow these steps:

- Pre-processing.
- Contour correspondence.
- Point correspondence.
- Triangulation.

Point-cloud surface reconstruction methods, however, are more general and do not tend to follow stages of reconstruction in this manner.

The proposed technique implements contour correspondence, point correspondence, and triangulation stages. There is no contour interpolation or other pre-processing steps performed on extracted contours.

3.2.1 Data Constraints

Source data is expected to be in a specific format. The constraints on data require that contours are an ordered set of points and that this order does not change throughout the data set. The reason for this is that the proposed use of dynamic time warping to solve this problem intends to create an alignment of points using this ordering. Noise removal is usually performed when contours are extracted. Since the reconstruction covered in this thesis uses extracted contours as input, it is assumed that all points in contour data will be used in the final mesh. The constraints on input data are summarised as follows:

- Contours must all be ordered in the same direction.
- Noise removal has been performed on extracted contours.

3.2.2 Pre-processing

Some contour-based reconstruction techniques have proposed the use of interpolation between slices [10], [27]. These algorithms aim to resolve issues such as branching by using interpolation to generate intermediate slices. The introduction of contour interpolation in large data sets can lead to processing and reconstruction of a model that contains a considerably large number of vertices and primitives.

3.3 Contour Correspondence

This section outlines the proposed rule-based contour correspondence technique used for reconstruction. A justification for the set of constraints imposed on contour correspondence is also given. Contour correspondence refers to the matching of contours based on some introduced constraints. As input, a set of n contours on the current slice, and the set of m contours on the neighbouring slice is given. Some pairing of contours between these two slices must be generated. The output

of contour correspondence can then be used to correspond points and then form a triangulation. Several cases exist in contour correspondence:

- No correspondence: A contour is found to match to no other contours.
- One-to-one correspondence: In this case, a contour a on the current slice is matched to a single contour b on the neighbouring slice. This contour b must also correspond to only a.
- One-to-many correspondence: A single contour on the current slice is matched to many contours on the next slice.
- Many-to-one correspondence: At least two contours on the current slice are matched to a single contour on the next slice.
- Many-to-many correspondence: In more complex situations, a correspondence may be generated with at least two contours on the current slice, and at least two contours on the next slice.

Many-to-many correspondence would ideally not occur, as it is considerably more challenging to create a triangulation for such a complex structure. The surface reconstruction method proposed does not aim to cover this case, but does aim to cover all other situations encountered in contour-based reconstruction.

There is another case which should be discussed relating to contour correspondence. When a contour is fully contained within another contour, it is referred to as "nested". For example, a slice of an HRCT stack may contain contours corresponding to airways nested inside the contour representing the lung boundary. In the point-correspondence stage of reconstruction, it is assumed that nested contours are either not present, or that they are detected during the contour correspondence phase and handled appropriately. Figure 3.3 shows how a nested contour may occur.

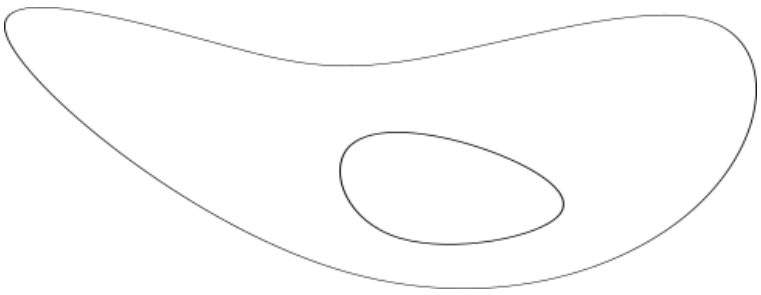


FIGURE 3.3: An example of a nested contour.

3.3.1 Rule-Based Correspondence

In the surface reconstruction method proposed in this thesis, contours are matched using a rule-based technique. This technique relies on assumptions made about the expected contour size and distance between contours. The contour centroid is given by the mean position of all points in the contour. This is given in Equation 3.1.

$$centroid = \frac{\sum_{p \in C} p}{n} \tag{3.1}$$

Where C is a contour of size n, and p is some point in C. From the centroid, the maximum distance to any point from the centroid is computed. The maximum distance from a contour centroid d_c is added to the maximum distance from a contour centroid to any point d'_c. If the distance between the two contour centroids (in the x and y dimensions only) is below this maximum, contours d_c and d'_c correspond to each other. The process of rule-based contour correspondence is summarised as pseudo code in Algorithm 14.

Algorithm 1: Centroid-based contour correspondence.

```
1  function ContourCorrespondence (slice_1, slice_1)
   Input  : Two sets of contours slice_1 and slice_2
   Output: A contour correspondence. That is a mapping between contours on
           slices.
2  Initialise correspondence
3  for Each contour in slice_1 do
4      for Each neighbour in slice_2 do
5          centroid_1 ← computeCentroid(contour)
6          centroid_2 ← computeCentroid(neighbour)
7          maxDistance_1 ← maxdistance(centroid_1, contour)
8          maxDistance_2 ← maxdistance(centroid_2, neighbour)
9          if centroidDistance ≤ (maxDistance_1 + maxDistance_2) then
10             correspondence.append(contour, neighbour)
11         end
12     end
13 end
14 return correspondence
```

An issue with this matching based on position is that small contours nested within larger contours may be matched incorrectly. To cover the nesting case, a size

threshold can be introduced for matching contours. Since bifurcation in medical imaging commonly results in volume divisions of around a half, contour sizes are expected to vary by around this much when branching cases occur [19]. However, branching can also occur in situations that are not a result of bifurcations. A case where a tubular structure bends back downward and is sampled twice is illustrated in Figure 3.4. The same tubular structure will be sampled twice at slices S_1 and S_2 but only once at S_3. A similar sampling is later shown in Figure 5.6.

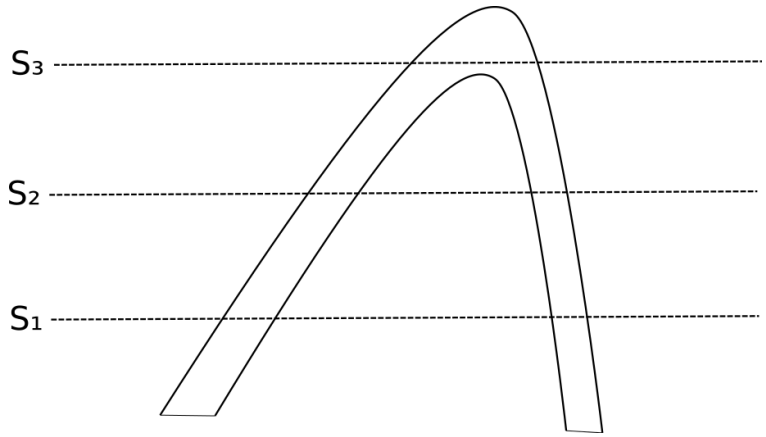


FIGURE 3.4: How a bending structure can generate a branching case.

Restricting the contour size to be too similar will result in branching cases being omitted. However, allowing too large of a contour size difference will introduce nested contours into the correspondence. In the proposed method of contour correspondence, both the contour centroid distance matching and contour size threshold are used to match contours. The combination of these two techniques both matches contours within a close enough proximity to each other, and within a certain size of each other.

3.4 Point Correspondence

The following section outlines the proposed use of DTW for point correspondence in surface reconstruction. Benefits from the use of DTW and its constraints are also explained. Point correspondence refers to the problem of deciding which points on a contour correspond to points on a neighbouring contour. In reconstruction, noise removal is assumed to have been performed when contours are extracted. All points in all contours are assumed to be a referenced vertex in the final surface. That is, all

points must be part of a triangle or face in the output mesh from reconstruction. Point correspondence identifies a set of edges from corresponding contours. These edges are also considered to be part of the final triangulated mesh. Some heuristic must be identified for matching points.

One method of doing this is by matching points based on minimum Euclidean distance, implemented by Mukundan [8]. Issues can arise with this solution when the contours start to become misaligned (contours are not directly above one another). Shortest path will match points closest to each other, which in the case of misaligned contours would not provide a good representation of the original surface.

3.4.1 Dynamic Time Warping

DTW can be used to match points, finding an optimal alignment between two series of points. This is outlined in Section 2.4. Euclidean distance between points on neighbouring contours is used as cost values for the proposed technique. With DTW these cost values are used in the generation of a warping path. This warping path will correspond to matched vertices on neighbouring contours. This optimal alignment will have the lowest total cost (assuming no use of global constraint). This means that the total length of all edges between contours will be minimised. Use of dynamic programming for reconstructing parts of meshes has been proposed in the past [26]. DTW possesses some useful properties for finding alignment between points in contours, as it ensures that the order of points is maintained. The following properties of DTW are useful for finding a point correspondence:

1. Monotonic condition: The alignment is monotonically increasing.
2. Continuity condition: The path advances one step at a time.
3. Boundary conditions: The first elements in sequences are always matched to each other; the last elements in sequences are always matched to each other also.

Additionally, the following optional constraints for DTW are available:

- Adjustment Window Condition: global restrictions on the warping path.

- Slope Constraint Condition: a restriction on the slope of the warping path.

The surface reconstruction technique proposed considers every point in a contour as part of the final reconstructed mesh. The first two properties are useful for ensuring that every point on the current contour is included in the alignment.

The Monotonicity Constraint

The monotonicity constraint is useful for finding an alignment for point correspondence. Since this constraint ensures that the indices for contour elements are always increasing, this aids in avoiding self-intersecting triangles. These self-intersecting triangles can still happen in the later triangulation stage.

The Continuity Constraint

By omitting sample points in a reconstructed mesh, real information about the source data is omitted from the surface. For this reason, the proposed method aims to include as many sample points as possible in reconstruction. The continuity constraint restricts index increases to at most one. Along with the boundary condition, this forces the warping path to include every point in both contours.

A combination of the monotonicity and continuity constraints can also prevent issues that occur with thin sections of contours. For example, consider a contour shaped similar to the Figure 3.5. Shapes similar to this are observed to happen near branching contours in real data. Monotonicity and continuity constraints ensure that the path will follow the contour around the thin segment correctly, instead of corresponding close points that are actually on opposite sides.

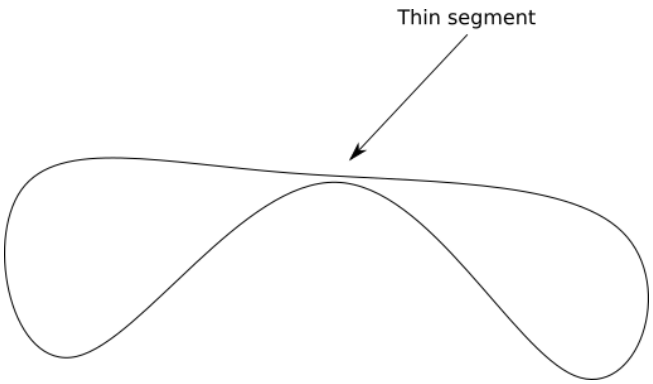


FIGURE 3.5: Contour containing a thin segment.

Setting the Boundary Conditions.

The boundary conditions in DTW mean that the first elements of feature sequences will be matched to each other in the warping path. Similarly, the last elements in both feature sequences must be matched to each other in the alignment. Assuming that the first points in corresponding contours will always match to each other can cause issues. The result of making such an assumption can cause issues in implementation for contour alignment. For example, consider the case where the first element in a contour $a_0 \in A$ and the first element in its corresponding neighbour $b_0 \in B$ are the pair of points with maximum distance:

$\forall a_i \in A, \forall b_j \in B, distance(a_0, b_0) \geq distance(a_i, b_j)$.

Use of such an initial boundary condition will generally result in bad overall alignment. To mitigate this some method to find an appropriate initial condition must be defined. Since the contours can be defined by a circular order, this also defines the end boundary condition as well. For the initial boundary condition, the implemented technique matches points with minimum x values from each contour: $a_{xmin} \in A$ and $b_{xmin} \in B$. Contours are then re-ordered so that these points a_{xmin} and b_{xmin} are the first features in each series while preserving the cyclic order. With this selection of more appropriate first (and last) indices, the boundary conditions become more useful.

Adjustment Window Condition

The implementation of DTW with no global constraints can result in misalignment where a small series of points are mapped from or to a large series. Global constraints can be used to prevent such alignments from happening. The Sakoe-Chiba constraint can be introduced to restrict the distance that the warping path can stray from the diagonal of the DTW matrix. Such a global constraint would also, in turn, restrict the size of sections of feature sequences that can be matched to each other. That is, a section of features points must be sufficiently similar in size to another for it to align. The current implementation of the proposed technique does not include any global constraints, but it is important to consider their use.

Choice of Cost Function

Creating the DTW table requires an objective function. For the problem of surface reconstruction, feature sequences consist of a set of points. One such goal is to minimise the total distance between points. The cost function for feature sequences to achieve this is the Euclidean distance between pairs of points. That is, our cost function for two points x and y becomes:

$$cost(x, y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{3.2}$$

3.4.2 Computing the DTW Array and Warping Path

Recall the recurrence relation in Equation 2.2.

$$dtw(i, w) = c(i, j) + min \begin{cases} dtw(i-1, j) \\ dtw(i, j-1) \\ dtw(i-1, j-1) \end{cases} \tag{2.2 revisited}$$

The recurrence relation can be used to generate a $n \times m$ array which contains all possible alignments of the feature sequences X and Y. In Algorithm 2, the dynamic time warping array is computed. The DTW array created in Algorithm 2 is then used to compute a warping path. A warping path with no adjustment window conditions or slope constraints can be found using Algorithm 3.

3.5 Triangulation

The final stage of reconstruction with the proposed technique is triangulation. This section elaborates how the warping path produced by DTW is used to generate an output mesh. Some approaches do not have a point matching step and go straight to triangulation from a set of corresponding contours [26]. For techniques which include point correspondence, the problem is posed as follows: Given two ordered points sets and a set of edges between these points, generate a surface which contains these points and edges. The proposed approach to this problem performs triangulation similarly to the approach by Mukundan [8]. The warping path obtained in the

Algorithm 2: DTW matrix computation.

Input: Two feature sequences, in this case contours, X and Y
Output: A DTW array.

```
function dtwArray (X,Y)
m ← size(X)
n ← size(Y)
Initialize the dtwArray to size m * n with all values set to 0
for i ← 1 to m do
    dtwArray[i, 0] ← ∞
end
for j ← 1 to n do
    dtwArray[0, j] ← ∞
end
dtwArray[0, 0] ← 0
for i ← 1 to m do
    for j ← 1 to n do
        cost ← objectiveFunction(X[i], Y[j])
        dtwArray[i, j] ←
          cost + min([dtwArray[i − 1, j], dtwArray[i, j − 1], dtw[i − 1][j − 1])
    end
end
return dtwArray
```

point correspondence phase is a series of pairs of points. Each step in the warping path corresponds to one of three cases:

- Neighbours p and q match to points p' and q' which are directly beside each other on the next contour.
- p matches to a series of at least two points $(p'_0, ..., p'_m)$ which may have any number of points between them.
- A series of at least two points $(p_0, ..., p_n)$ correspond to a single point p' on the next contour.

For triangulation contour-clockwise winding order is used. The first case is trivial. A quad can be generated $Q = (p, q, q', p')$, or two triangles $T_1 = (p, q, q')$, $T_2 = (p, q', p')$. For the many-to-one and one-to-many cases note that these parts of the surface are representable as triangle fans. In the second case, a triangle fan can be generated using p as the central vertex and continues from p'_0 through any extra vertices to p'_m. Note that if there are only two points matched to p, then a triangle $T = (p, p'_m, p'_0)$ can be created. Similarly, for the third case, a triangle fan is

Algorithm 3: DTW path calculation.

Input: An $m \times n$ dynamic time warping array dtw
Output: Warping path

```
function dtwWarpingPath (dtw)
Initialise warpingPath
i ← 0
j ← 0
warpingPath.append((i, j))
while i < m − 1 do
    while j < n − 1 do
        // Get the indices of the next cell with minimum cost.
        i, j = getMinIndices(dtw[i-1, j], dtw[i, j-1], dtw[i-1, j-1])
        warpingPath.append((i, j))
    end
end
return warpingPath
```

created which starts with the central vertex p' and continues through the vertices $(p_0, ..., p_n)$.

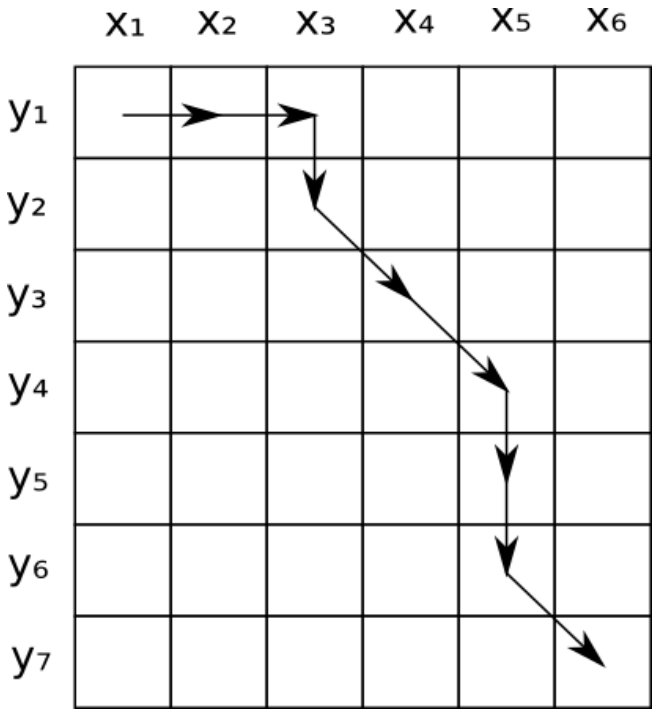


FIGURE 3.6: Warping path between two sequences.

Consider the warping path between two sequences $X = (x_1, x_2, \ldots, x_6)$ and $Y = (y_1, y_2, \ldots, y_7)$ in Figure 3.6. To triangulate this series the warping path is treated as a point correspondence. The triangulation starts by generating a triangle fan with y_1 as the central vertex, and x_1, x_2, x_4 as the other vertices. A triangle x_3, y_1, y_2 is then

generated. This is followed by two quads (four triangles) comprising of x_3, y_3, y_4, x_4 and x_4, y_4, y_5, x_5. Another triangle fan is generated with x_5 as the central vertex with y_4, y_5, y_6. Lastly, another quad is generated with vertices x_5, y_6, y_7, x_6. The use of triangle fans reduces the number of stored indices for the mesh. They can be re-ordered, or re-structured, before rendering to take advantage of optimisations such as cache locality.

Figure 3.7 shows the result of primitive generation. Note that this is not strictly a triangulation as quads are left for demonstration, but this would be represented as two triangles in implementation. Another thing to note is that the final vertex is the same as the initial one in implementation. This property is because of the circular order of the contour. A hole will be left near the end of contours if the method is not implemented this way.

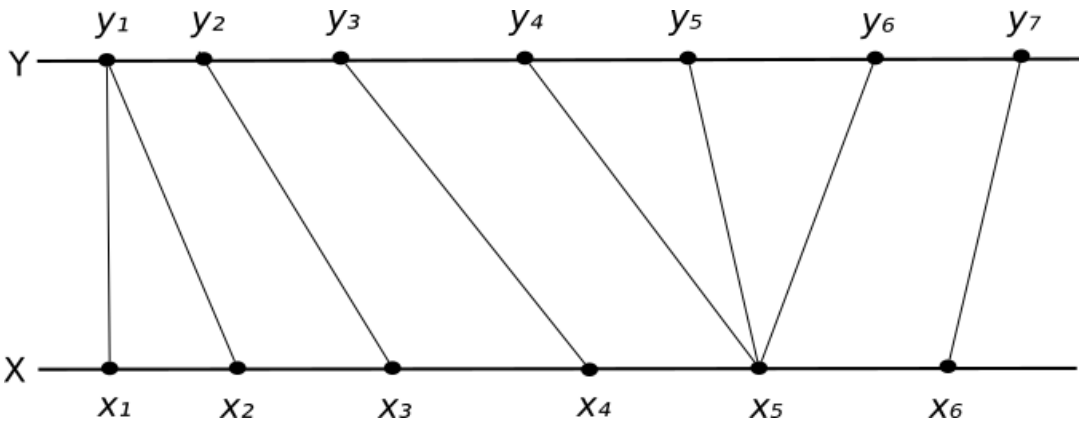


FIGURE 3.7: Primitive generation from point alignment.

For the one-to-one case of point correspondence (diagonal warping path step), note that the number of triangles generated for each step is two. For the other two cases, one-to-many and many-to-one, each step in the warping path generates a single triangle. Given feature sequences of length m and n, then using the boundary conditions of the warping path, the initial position of the path is $(0, 0)$ and the final boundary condition is (m, n). The number of triangles generated for two feature sequences of size m and n is the L_1 distance from $(0, 0)$ to (m, n). If there is a contour C_1 with m points that is found to correspond to another contour C_2 of size n. Then $m + n$ triangles are generated between these two contours during reconstruction.

3.6 Handling the Branching Case

The branching condition refers to cases where contours are matched in a many-to-one (or one-to-many) scenario. A triangulation needs to be generated for slices which contain branches. This case is difficult because the generation of triangles is not as trivial as it is with one-to-one cases. If all points are matched on all contours to all points from corresponding contours on a neighbouring slice, then holes and self-intersections will be generated. This case needs to be treated differently with the proposed technique for triangulation. This section explains how the proposed technique generates triangulations in the case where a branching structure is encountered.

3.6.1 Contour Merging

For point correspondence to be accurately performed in the branching case, DTW must be used to generate an alignment for all branching contours. A technique is proposed for joining these contours logically, allowing for a point alignment which includes all branching contours. Joining of branching contours allows them to be treated as a single feature sequence. This merged contour can be used in the DTW array computation and so a point correspondence can be found which considers all points in branching contours. The ordering of points when joining contours needs to be considered, as the alignment created by DTW uses this ordering. The process for this is detailed below, with edge case considerations explained.

A simple branching case is considered. In this case a single contour $B_c = [b_1, ..., b_m]$ is on the current slice, and two branching contours on the next slice $C_c = [c_1, ..., c_n]$ and $D_c = [d_1, ...d_k]$. It is assumed that the contour correspondence stage has established that both C_c and D_c are matched to B_c. This configuration is a one-to-two branching case, common in medical imaging with a sampling of bifurcating structures.

The closest pair of points (c_i, d_j) can be found using Euclidean distance where $c_i \in C_c$ and $d_j \in D_c$. These points are later referred to as "merge vertices". Using these two closest points contours are re-ordered so that c_i and d_j are now the start and end of their contours C_c and D_c respectively. That is, $[c_1, ..., c_i, ..., c_n]$ becomes

$[c_i, ..., c_n, c_1, ..., c_i]$. If $i = 1$ then the first element is simply inserted at the end. Similarly, if $i = n$ the last element is inserted at the start of the contour. Once these contours are reordered, they can be merged together by appending one contour to another. The merged contour is now defined by $M_c = [c_i, ..., c_n, c_1, ..., c_i, d_j, ..., d_m, d_1, ..., d_j]$

Algorithm 4: Contour merging procedure pseudocode.

```
1 function mergeContours (A, B)
  Input  : Two contours A and B
  Output: A single contour containing all points from A and B.
2 (i, j) ← closestPairIndices(A, B)
3 A_r ← reorderContourFromIndex(i)
4 B_r ← reorderContourFromIndex(j)
5 M ← joinArrays(A_r, B_r)
6 return M
```

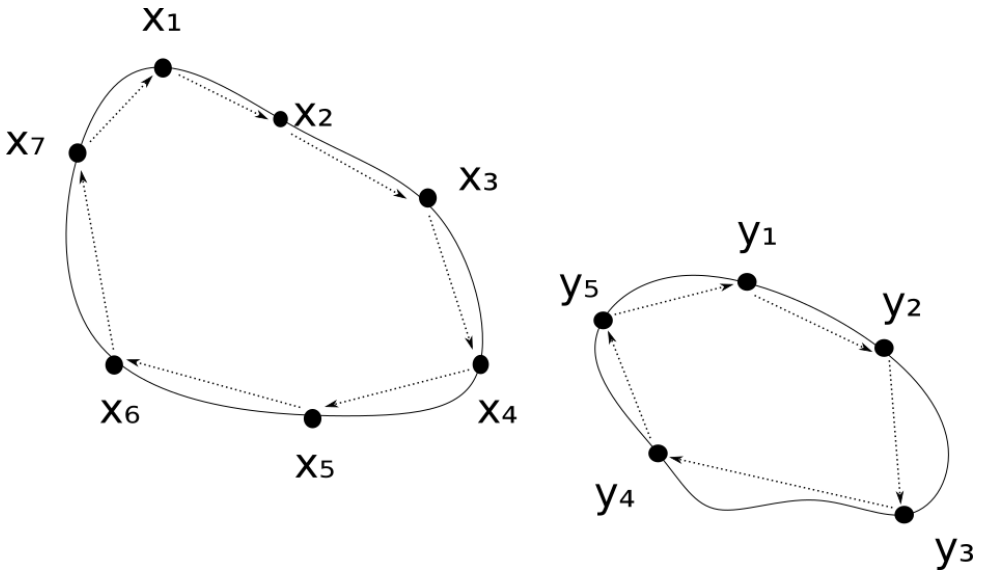


FIGURE 3.8: Contours before merging. Assume x_4 and y_5 are the closest points.

Figure 3.8 shows two contours on the same slice $X = [x_1, x_2, x_3, x_4, x_5, x_6]$ and $Y = [y_1, y_2, y_3, y_4, y_5]$. Assume in this case that they have been matched to a single contour on another slice. To merge these contours, begin by finding the closest pair of points from X and Y and reordering as summarised by pseudo code in Algorithm 4. The closest pair of points in this example is x_4 and y_5. Contours can be re-ordered and combined as described above. The resulting merged contour is given by $M = [x_4, x_5, x_6, x_7, x_1, x_2, x_3, x_4, y_5, y_1, y_2, y_3, y_4, y_5]$. The result of this merging process is shown in Figure 3.9. This merging process allows the DTW algorithm

to consider points from all contours in a correspondence when creating an alignment.

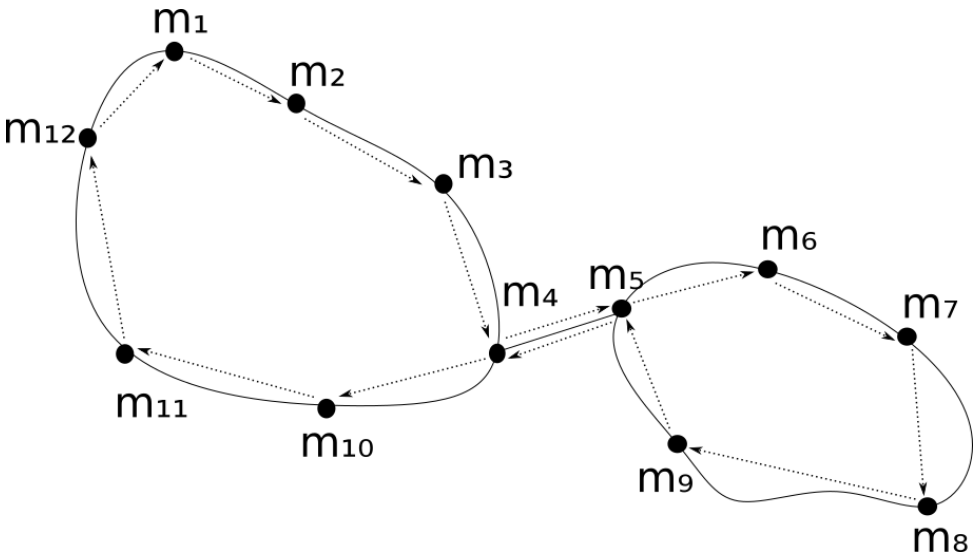


FIGURE 3.9: Contours after merging.

The introduction of an ordering of all points in branch contours partially mitigates this problem. By merging contours that branch (treat them as one contour), DTW can be used to find an alignment which includes all these branching contour points. From this alignment, point correspondence and triangulation can be performed almost the same way as in the one-to-one case. If the merged contour M_c is treated as if it is a single contour and perform point correspondence from B_c to M_c the produced surface will be mostly correct. However, this results in the erroneous generation of two extra triangles and two holes in the mesh.

3.6.2 Handling Erroneous Triangles

The two extra triangles are generated where the two contours were merged. The triangles generated will both include the closest two points originally used for merging the two contours. The third vertex in this erroneous triangle will be some matched point on the contour that is corresponding to the merged contour. To avoid these erroneous triangles, any triangle that includes both of the merge vertices can be omitted. When creating a mesh from the warping path, any triangle T that satisfies the following is omitted:

$$o_1, o_2 \in T$$

Where o_1 and o_2 are the closest two vertices between the contours. For example, in Figure 3.9 any triangle that contains both vertex m_4 and m_5 would be omitted during triangulation.

3.6.3 Patching Mesh Holes

Unfortunately, two holes are still left in the output mesh. Patching of these holes is a slightly more complicated issue than handling erroneous triangles. Consider the first merging point m_1 encountered during reconstruction. This point is encountered twice during point correspondence. The first time it is matched to at least a single point c_{i+k} as in Figure 3.10. Note that there may be more points matched to m_1; however, c_{i+k} is an important vertex as this is the last vertex encountered before the point correspondence moves to the other contour. When m_1 is reencountered during point correspondence, it is when the point correspondence moves back to this original contour. m_1 is again matched to at least one point c_j and may be followed by any number of points. The first hole generated in the branching case corresponds to the triangle (m_1, c_j, c_{i+k}) (assuming counter-clockwise ordering). By adding a triangle (m_1, c_j, c_{i+k}), this hole is patched.

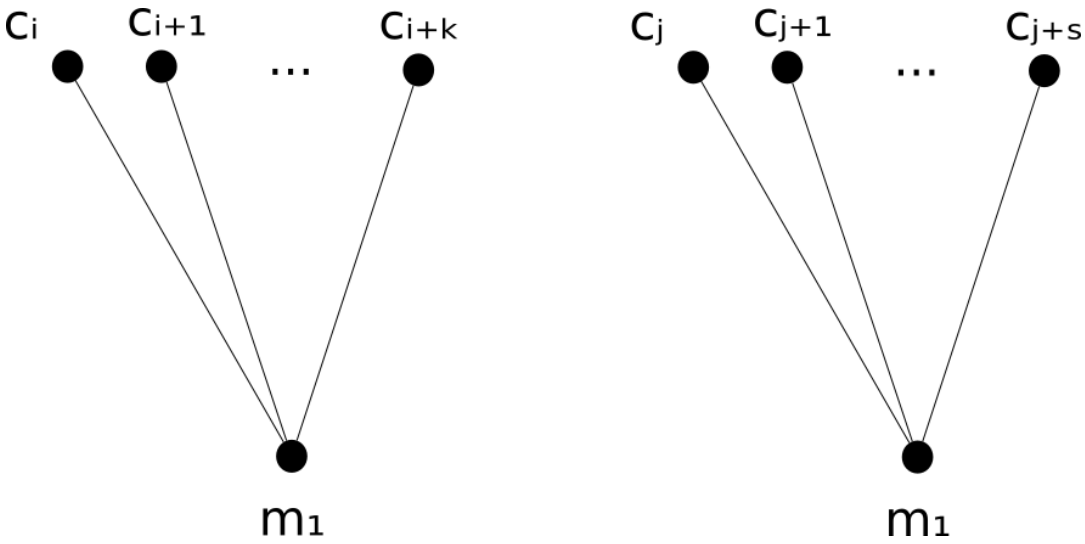


FIGURE 3.10: Handling the first merging point generated hole.

The second merging point encountered is also part of a hole created in the mesh. Similar to the first merging point, this point will also be encountered twice in the

warping path. The hole is affected differently because of the way this correspondence is established. When processing the second hole, this order is reversed. Figure 3.11 illustrates an example of point correspondence for the second merging point. The hole in this case consists of m_2, d_i (the first point corresponding to m_2) and d_{j+s} (the last point corresponding to m_2). This hole can be patched by adding the triangle (m_2, d_i, d_{j+s}).

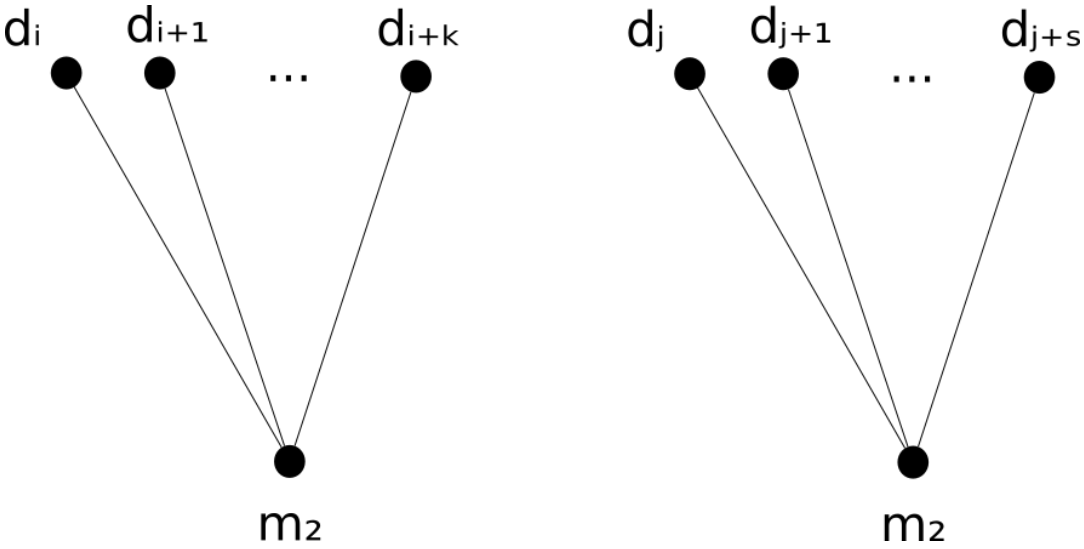


FIGURE 3.11: Handling the hole generated near the second merging point.

Note that this requires the ordering of points to be in a specific direction. This technique will not work if the order of contours on the same slice is different. It is worth noting that the DTW algorithm, in general, will have issues if the direction of contour ordering is not held constant. Both patch triangles should be flipped in the case where the contour points are ordered in the opposite direction, as this will result in back-facing triangles. However, the current implementation of this technique only works in one direction.

Multiple Branching.

In the case of one-to-many or many-to-one branching where there are more than two branches, contours can be merged iteratively. Two contours can be selected to be merged. Then merge this new contour with the remaining contours as needed. Note that merge vertex indices must be stored so that extraneous triangles can be

omitted and to patch the holes produced by the merge. The technique in its current state does not support handling of many-to-many contour correspondences.

3.7 Post-processing

This section briefly covers how surface smoothing can be used to process a reconstructed mesh for greater visual quality. Considerations for preserving mesh data are also mentioned.

3.7.1 Surface Smoothing

FIGURE 3.12: Comparison between smooth and flat shading for reconstructed models.

Surface smoothing can create visually smoother meshes. A potential issue for this in surface reconstruction is that some surface smoothing techniques modify vertex positions. Since these vertices are parts of the original surface, smoothing the surface modifies real data. Despite the smoothed mesh being potentially more visually

appealing, this may be unwise to discard important data during reconstruction. Subdividing and smoothing new vertices may work, but introduces extra vertices to an already large model. Use of smooth shading is recommended as it does not modify the reconstructed model and can still provide a visually appealing rendering. The visual difference between flat-shaded and smooth-shaded models can be seen in Figure 3.12. Use of vertex normal vectors instead of face normal information will further improve the visual quality of rendering.

3.8 Summary

This chapter provided details for the proposed technique of contour-based surface reconstruction. This technique is founded on the use of DTW for the alignment of points between contours. The reasoning for the use of DTW stems from the fact that contours are often similar in shape, but will be offset from each other in the x and y axis. A rule-based contour correspondence technique is detailed which allows for full contour-based surface reconstruction. Edge cases such as branching are handled with a contour merging technique to ensure that alignments created with DTW consider points in all corresponding contours. The main improvements that the proposed technique provides over previous efforts is the ability to create alignments for branching contours, and for contours which are similar, but significantly offset from each other. The implementation of the proposed method can create logical triangulations for branching structures when the correct contour correspondence is obtained.

Chapter 4

Ground Truth Generation

This chapter describes the methods and test models designed for evaluating the performance of reconstruction algorithms. To provide a comparative analysis of accuracy between surface reconstruction techniques, visual inspection of real data can be used. Ground truth model comparisons are used to obtain quantitative analysis about the accuracy of reconstructions. This thesis proposes a technique that can be used for generating test data sets for contour-based surface reconstruction. Section 4.1 explains the process and purpose of ground truth testing. Section 4.2 provides a summary of the test meshes generated for testing reconstruction, how they were generated, and justification on why these meshes were selected. Following this, Section 4.3 proposes a process for generating test contours from piecewise-planar surfaces. An alternative technique for generation of test contours is briefly covered in Section 4.4, though this method only allows for visual analysis.

4.1 Ground Truth Testing

Ground truth meshes are used in research to compare the accuracy of surface reconstruction techniques. Measurements such as Hausdorff distance cannot be used with real data as original surfaces are unavailable. Only a sample of the surface is given in the form of contours. In both contour-based surface reconstruction and point-cloud surface reconstruction, it is common to generate ground truth meshes and produce sample data [26], [62]. Since the original mesh is available in ground truth testing, reconstructions can be directly compared. The process of ground truth testing is shown in Figure 4.1.

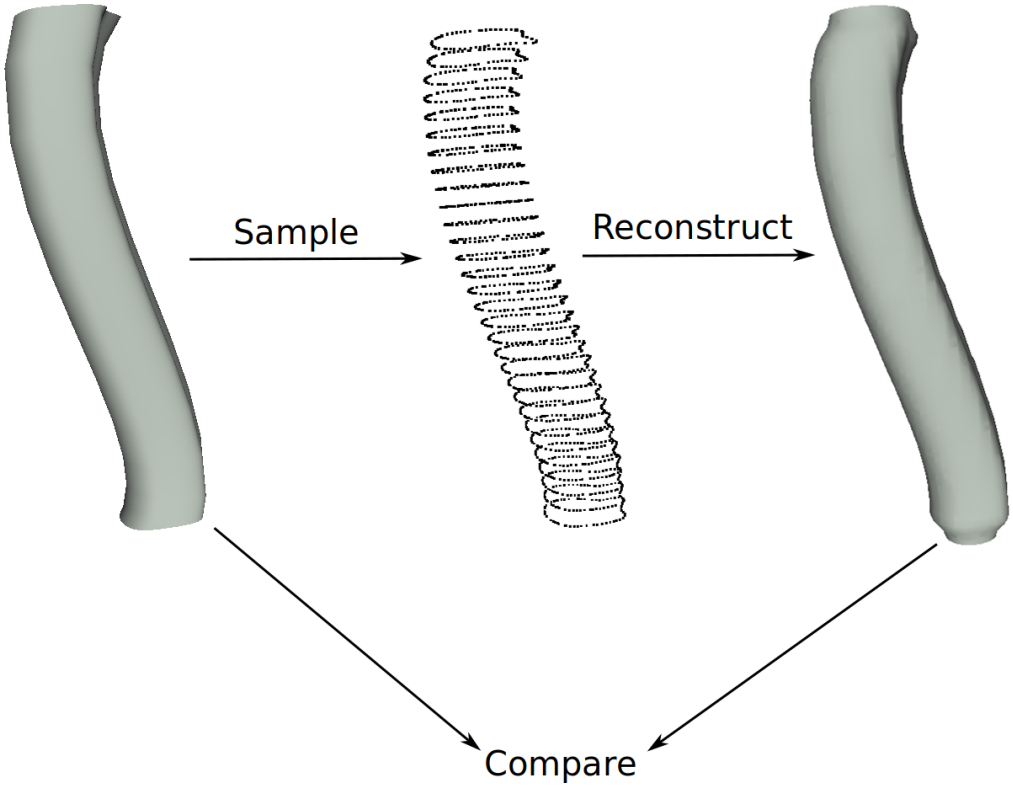


FIGURE 4.1: Ground truth testing process

Ground truth meshes which simulate common scenarios known to cause issues in contour-based surface reconstruction are explored. By replicating these specific configurations, techniques can be explicitly tested for accuracy in contour-based reconstruction. These tests are designed to evaluate point alignments created by DTW for applications in medical imaging. Branching is a commonly encountered issue in contour-based surface reconstruction for medical imaging, and there is a sizeable amount of research that considers this problem [8], [10], [26]. With respect to contour-based surface reconstruction techniques, meshes that involve branching structures provide test cases for contour correspondence that is not one-to-one. It is worth noting that preliminary test meshes are not designed to prove accuracy in general for a reconstruction technique. They are included to provide analysis of common scenarios that may be encountered specifically in the application of medical imaging. This test-driven approach allows for the development of reconstruction techniques before real data is accessible.

For testing point correspondence, this thesis aims to also include meshes with offset contours. A mesh that is trivial to reconstruct is one where a single contour

will be the same contour located directly above another contour. Simply matching all points with the shortest path would produce a visually appealing mesh in this case. However, it is unlikely that the complex structures in a real medical imaging scenario would all be aligned with the medical imaging device in such a way that produces aligned contours. For this reason, test data is produced which contains contours that are offset between slices from one another. Such a data-set would provide an evaluation of alignments produced by point correspondence techniques on offset contours. Figure 4.2 shows an example of what such a set of contours may look like. Matching points on these contours becomes more difficult with such a configuration.

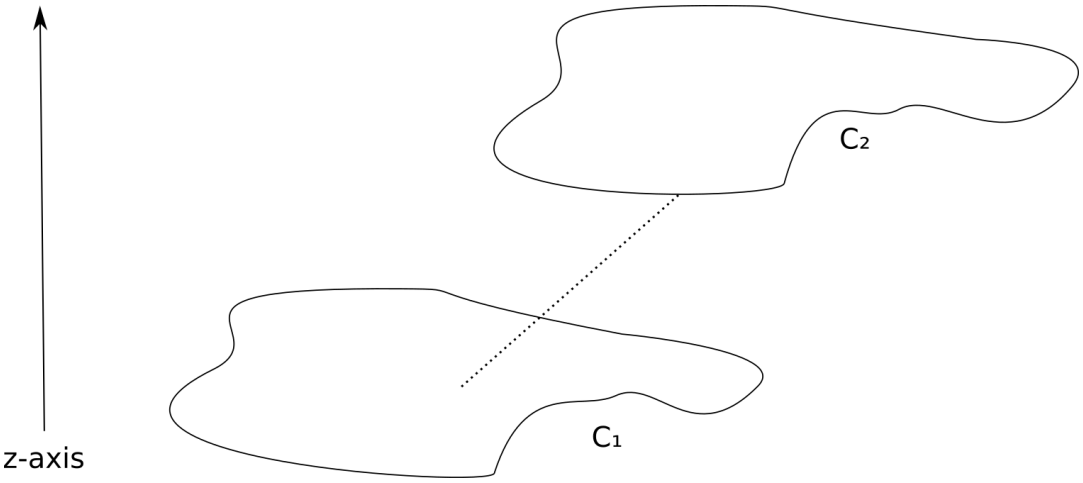


FIGURE 4.2: Contours that are similar but offset from each other in x and y axis.

4.2 Test Models

This section describes the selection of test models and justification for their inclusion in the analysis. Blender3D [83] was used for the creation of all test models in this section. Most surfaces were formed by creating a surface of revolution around a Bezier curve. For branching structures, more than one Bezier curve was used to generate surfaces of revolution. Four ground truth test models are created for analysis, each serving a unique purpose in testing. All models were created with the intention of emulating issues that were observed in real data. This section justifies

their use and explains their relevance towards contour-based surface reconstruction in medical imaging.

4.2.1 Model Creation Process

There has been significant work in the generation of models of bronchial structures [19], [70], [71]. These generated models were not intended for use in testing, but they are worth considering for this application. Although model generation like this would be useful for testing most cases, Blender was chosen instead as it allowed us to quickly create models for specific cases which were observed to cause issues in reconstruction.

The general process for creating models was to create tubular structures using the Bezier curve function. The way this surface is created around the Bezier curve can be modified with the use of a bevel object. Bezier curves were chosen because of the relatively smooth surfaces that they generate. Boolean polygon operations available in Blender were used to join these tubular structures without introducing unintended internal structure. If greater mesh quality is required, Blenders "subdivide and smooth" operation is used to create a smoother mesh. Although this process is straightforward, it allowed for the quick creation of a significant amount of test cases.

4.2.2 Test Models

Test models are generated to evaluate how a surface reconstruction algorithm responds to specific scenarios encountered in medical imaging. The main contribution of this research is to provide a technique for corresponding points that can handle branching contours. For this reason, test models containing bifurcating structures are generated. One such example of a branching structure that is commonly encountered in medical imaging is bifurcations of the trachea and the subdivisions of bronchi that follow [19]. Another example where a branch is encountered is when a tubular structure bends back downwards and so branches into two contours (Figure 4.4). Although this is not a case where an actual bifurcation takes place, it still

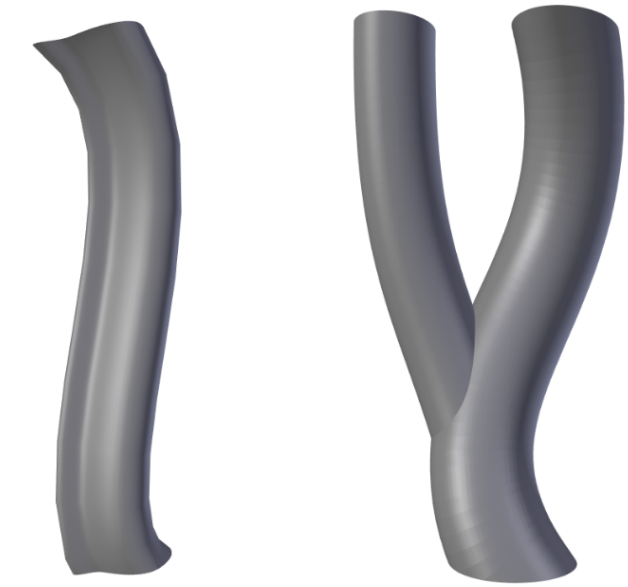
results in a branch. Models are included which aim to provide mock data for both of these cases.

A simple case with no branching is included for early testing. This case consists of a single tubular structure generated from a surface of revolution. The purpose of these test cases is to evaluate how an algorithm will function in a trivial case. Finally, test models with multiple branches are generated to evaluate how algorithms handle more complex structures. How these surface reconstruction algorithms handle real data should be represented accurately with the way they handle test data.

Simple Bent Tube

The simplest test case involves no branching and little variation between slices. This test model is included for demonstrating that the reconstruction algorithm can function in the simplest cases. This model is a simple tubular structure that is created in Blender by generating a surface around a Bezier curve. A simple test mesh such as this can show weaknesses in point correspondence and triangulation methods. Figure 4.3a shows a rendering of this source test model before sampling. This test case does not have any branches, and there is only one contour per slice (one-to-one contour correspondence).

Simple Branching Structure



a) A simple bending structure b) A simple branching case

FIGURE 4.3: Simple ground truth testing models.

The next case included is a model of a single branch (Figure 4.3b) which is common in medical imaging as a bifurcation. Since this model is created explicitly to test only the branching case, there are little extra features to this structure. This model was created in Blender using two tubular surfaces modified by Bezier curves. These two curves were then joined with a Boolean intersect to create an internal structure that resembles an anatomical structure. Contours on this surface are sampled in such a way that is one-to-one aside from correspondence between two specific slices at the branch. These slices will have a one-to-two or two-to-one correspondence depending on how the structure is sampled.

Non-Bifurcation Branching Structure (Bent Tube)

Another edge case that creates a one-to-two correspondence involves a tubular structure that can be sampled twice as it bends back downward in the z-axis. Figure 4.4 demonstrates a test mesh where this occurs. Unlike a regular bifurcation in medical imaging, this type of branching results in contour sizes that are much less predictable (see also Figure 5.6). For example, a tubular structure with a long bend and relatively

small volume can result in two small contours suddenly merging into a much larger contour.

FIGURE 4.4: A test model that generates a branch-like configuration of contours due to bending.

Multiple Branching Structure

The final test model used for ground truth testing was created with multiple bifurcations. The model itself was created in a similar manner to the simple branch in Figure 4.3b. In bronchial structures, it is observed that the volume of structures decreases by half in the case of a bifurcation [19]. For this reason, this model includes a sequence of branches that gradually get smaller. This test highlights the resilience of a reconstruction technique where the level of detail given in contours is variable. Performing a good reconstruction of this model shows:

- The reconstruction technique can handle multiple branches.
- The reconstruction technique can work at a dynamic level of detail. That is, it can generate meshes well in cases the number of points per contour is variable. This property is especially relevant for reconstruction techniques that require manual input or configuration to function correctly.

- The reconstruction technique can perform some form of contour correspondence in both directions.

FIGURE 4.5: Test model with multiple branches

4.3 Contour Generation

Blue noise sampling is a technique that can be used for uniformly sampling points on a mesh. This method can be used for the creation of data sets for testing point-cloud based reconstruction. However, many contour-based reconstruction techniques rely on additional information provided by a contour, such as the ordering of points. For this reason, this thesis aims to generate contours from the ground truth models created, instead of generating unordered point sets. Thus, this introduces another problem for the generation of test contours that contain sufficient information for contour-based reconstruction from piecewise-planar meshes. The solution proposed uses plane-triangle intersections to create a collection of intersection lines, which points are sampled from and then ordered into distinct contours. This section presents the solution to this problem in detail.

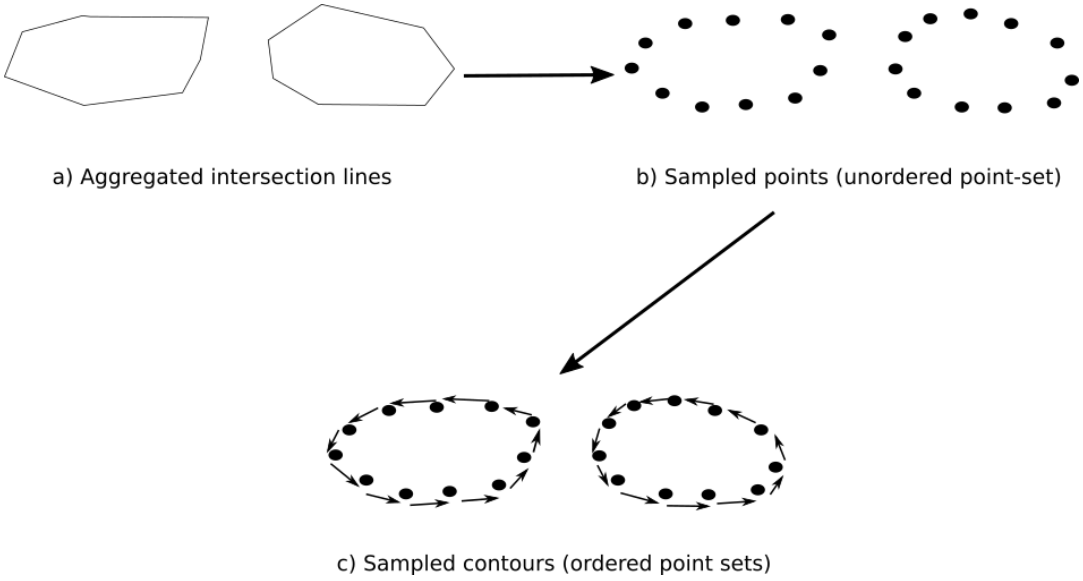


FIGURE 4.6: An illustration of the contour generation process

4.3.1 Problem Statement

This problem is posed as follows: Given a piece-wise planar mesh, generate a series of test slices on this mesh. Slices can contain a number of distinctly identifiable contours and are evenly spaced between layers. Test contours are defined as an ordered set of points. The following information must be available in the produced test contours:

- Generated contours on the same slice must be distinguishable from each other.
- All contours must be defined in a circular order.

These requirements allow for generated test data which reflects data representation given in real data. Note that these constraints distinguish this problem from generating test data in point cloud reconstruction. With a point cloud, the input data is simply an unordered set of points which are not restricted to fixed planes/slices.

4.3.2 Solution Summary

The contour generation method aims to create test contours for some ground truth mesh. To sample a single test slice on a contour, several steps were required.

- Find intersecting triangles of mesh with a given plane using fast intersection tests [84].
- Get intersection lines for intersecting triangles and aggregate them into a set.
- Sample points from intersection lines to create an unordered point set.
- Partition unordered point set into several point sets using a threshold related to point distance.
- Find the contour centroid and order points by their relative angle to this centroid.

The later stages of this process are illustrated in Figure 4.6. As input, the method takes a source model and a contour sampling distance. The contour distance is the distance between generated contours. Point sampling distance refers to the distance between sampled points in each contour.

4.3.3 Implementation Details

The first stage of generating samples is to find model extremities. The highest and lowest z values of any point in the model are found. The process starts by creating a test slice at a height h slightly above the lowest z value. This height is increased by the contour distance and re-sampled until it is larger than the highest z value. The test slice sampling process is explained below.

To generate a contour at a given height h the following can be done. Firstly, define a plane in Hesse normal form using point $Q = (0, 0, h)$ and normal $\vec{n} = (0, 0, 1)$. Plane-triangle intersection tests can then be used to determine which triangles in the mesh will be part of the contour. For plane-triangle intersection tests and computation of intersection lines, the method proposed by Möller [84] is used. The proposed contour generation method computes planes for both triangles and determines whether all points in a triangle lie on one side of a plane by checking the signed distance to that plane.

The signed distance is computed from the plane to each point on the triangle. If the sign for any of the three point-plane distances differs, then the points are on different sides of the plane, and so there is an intersection. For further clarity, an example of how this may be implemented is given in Algorithm 5. Equation 4.2 is used to compute the signed distance from a point to a plane. This equation assumes the plane is in Hesse Normal Form, given in Equation 4.1.

$$\vec{n} \cdot x = -h \tag{4.1}$$

where $\vec{n}$ is the plane normal, h is the distance from the plane to the origin.

$$distance(x_0) = \vec{n} \cdot x_0 + h \tag{4.2}$$

Algorithm 5: Fast intersection testing for triangle-plane intersections.

Input: A triangle, T
Input: A plane, P
Output: True if the triangle intersects the plane. False otherwise.

```
function TrianglePlaneIntersection (T,P)
p1distance ← distance(P, T.p1)
p2distance ← distance(P, T.p2)
p3distance ← distance(P, T.p3)
if p1distance > 0 and p2distance > 0 and p3distance > 0 then
    return True
end
if p1distance < 0 and p2distance < 0 and p3distance < 0 then
    return True
end
return False //otherwise.
```

Intersection testing as in Algorithm 5 is fast, but it does not give the intersection

lines themselves. For all of the intersecting triangles, plane-line intersections are calculated. If it is known that a triangle intersects a plane, exactly two edges e_1, e_2 of the triangle intersect with this plane [84]. The intersection point of a plane and edge $e = (p'_1, p'_2)$ can be computed as follows [85]:

$$p_{intersection} = p'_1 + t\vec{d} \tag{4.3}$$

where $\vec{d} = p'_2 - p'_1$, and

$$t = \frac{(Q - p_1) \cdot \vec{n}}{\vec{d} \cdot \vec{n}} \tag{4.4}$$

Note that this relies on the lines intersecting the plane, which is asserted by the previous fast intersection test. An intersection line is then given by the two intersection points which are computed from intersections of e_1 and e_2 with the plane.

By aggregating all intersection lines, a piecewise representation of an intersection with the test mesh is given at this specific height. Points are sampled on these intersection lines to create unordered test contours. Note that this is different from the contours in real data as these points will not be spaced at even distances in the x-axis and y-axis. In real data extracted contour points are represented as pixels in images, which have fixed sizes in the x-axis and y-axis. For testing these contours are sufficient.

All sampled points at the current height h are aggregated into an unordered point set. For point-cloud reconstruction techniques data is output in this format. However, contour-based reconstruction techniques rely on additional information in order to function, such as knowing which points are contained within a contour, and the order of points in a contour. Because contours are ordered point sets, the generated point sets must be ordered. Since the sampling distance for points is known, individual contours can be extracted from the unordered point set. The process starts by taking a point from the unordered point set G_A (without replacement) and adding it to a new point set G_i. Points that are within a close enough distance to any point from G_i are added to this set until only distant points remain in G_A. Once there are no points within the threshold, the unordered point set is checked for emptiness. If not empty, a point is chosen and a new contour created, following the same process

of adding points. Once there are no points left to process, a set of partitioned (but still unordered) point sets is returned.

To generate an ordering on the points in each partitioned point set, the centroid of these points is first computed. Then a vector $\vec{l} = (1, 0, 0)$ is defined and for each point p the angle from the vector defined by $centroid - point$ to $\vec{l}$ is calculated. Sorting this set of points by an angle is used to provide an ordered set of points.

This process of test slice generation is restated as pseudo code in Algorithm 6.

Algorithm 6: Contour generation process pseudocode.

Input: A set of triangles, V_T

Input: A plane, P

Output: A set of contours

```
1  function generateContours (P,V_T)
2  V_intersecting ← findIntersectingTriangles(P, V_T)
3  Lines_intersection ← {}
4  for Triangle in V_intersecting do
5      IntersectionLine ← computeIntersectionLine(Triangle, P)
6      Lines_intersection ← Lines_intersection ∪ IntersectionLine
7  end
8  UnorderedPointSet_global ← samplePoints(Lines_intersection)
9  PartitionedSets_unordered ← partitionPointSets(UnorderedPointSet_global)
10 TestContours ← {}
11 for PointSet_unordered in PartitionedSets_unordered do
12     ContourCentroid ← computeCentroid(PointSet_unordered
13     PointSet_ordered ← sortByAngle(PointSet_unordered, ContourCentroid)
14     TestContours ← TestContours ∪ PointSet_ordered
15 end
16 return TestContours
```

By allowing the contour distance to be varied, the sampling rate of the test set provided for reconstruction can be adjusted. Reconstruction algorithms can then be evaluated for ideal situations where there is a large amount of contour information provided and also situations where there is a low amount of sample points, which is often the case where algorithms fail or do not generate accurate meshes. From

observations of the real test data set, we find that the contours in this particular set usually consist of between 30 and 180 points. There are also many cases that have contours of as low as 10 points. For this reason, the generated test data sets generally contain a varied size of between 30 and 200 points per contour.

4.3.4 Limitations

The method used to order points uses the relative angle of points with a vector defined from contour centroid. With a convex shape this works fine, for more complicated contour shapes this can cause issues. Points may be ordered incorrectly with contour shapes that have large concavities. A more concerning case happens when the contour centroid is located outside of the actual contour. Unordered point sets will be created, and they will be partitioned into unordered contours. However, the ordering stage of this process will create an incorrect ordering of points. Currently, there is no implemented way of validating that contour points are ordered correctly. For the test set used, these limitations are avoided with the production of models containing convex structures with minor concavities.

4.4 Alternative Method for Test Generation

This section details another technique for rapid generation of contour test data. The method takes an input image which represents a branching structure and generates test slices from this. The main benefit of this technique is that it allows for the rapid generation of specific test cases without access to real test data. Though this technique does not allow for ground truth testing, it still shows benefits with the ability to produce user-defined test sets for visual analysis.

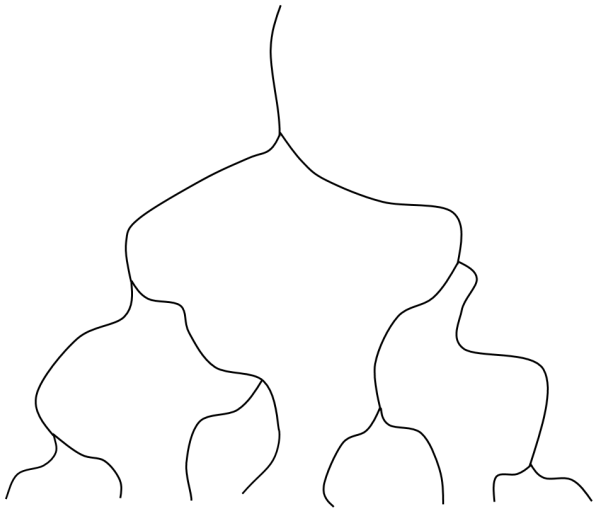


FIGURE 4.7: Side on drawing representing a branching structure.

This process takes an input grey scale image such as the one in Figure 4.7 and generates a set of test contours. As input, the program also uses an inter-slice gap. The grey scale image is read a row at a time, jumping forwards by the inter-slice distance. Where coloured pixels are encountered in the row, a contour is created. Each contour is created by generating sample points revolving around the coloured pixel. Lines are created between each point and the previously generated one, defining a closed contour. Noise is also created by randomly varying the distance from the coloured pixel to the sampled points. This process is summarised as pseudo code in

Algorithm 7.

Algorithm 7: Generation of contour test data from drawn images.

Input: A black and white image which contains a drawn branching structure.

Output: A test contour stack.

```
1  function generateContoursFromDrawing (image, intersliceDistance)
2  Slices ← {}
3  for row in image do
4      Contours ← {}
5      for pixel in row do
6          if pixel is coloured then
7              Contour ← generateContourAroundPoint(pixel)
8              Contours ← Contours ∪ Contour
9          end
10     end
11     Slices ← Slices ∪ Contours
12     row ← row + intersliceDistance
13 end
14 return Slices
```

Note that this can produce intersecting contours (especially near branching contours). To resolve this a line intersection test is used to find the two intersection points of the contours. The intersection points are then used to re-sequence the contours. This technique does not allow for ground truth testing as models are not generated from a ground truth surface. Test data generated by this method is still useful for visual inspection of models. This technique also only produces branches which vary in the x axis as the drawing can only represent structure in the x axis. The main advantage of producing test data with this technique is the ability to produce test contours rapidly. All that is required to create test data is an image editor which allows a user to draw a connected structure such as the one in Figure 4.7.

4.5 Summary

This chapter provides insight into the test data generation techniques used. The ground-truth test generation method allows for test contour generation from a piece-wise planar mesh. Ground truth testing is designed to be used with metrics such as Hausdorff distance, which provides quantitative metrics about distances between mesh geometry. A second test generation technique is also proposed in Section 4.4 which allows for the generation of test data from an image drawn with a basic image editing tool. The second technique exists only for visual analysis of reconstructions, as there is no ground truth. The use of both of these techniques allows for an in-depth analysis of how surface reconstruction techniques will handle specific cases that are expected to be encountered in medical imaging.

Chapter 5

Analysis

This chapter provides a detailed evaluation of the application of DTW for contour-based reconstruction. Additionally, a comparison with existing free and open source implementations of common surface reconstruction techniques is provided. Quantitative analysis is provided with the use of Hausdorff distance from comparison with ground truth surfaces. Section 5.1 provides the results of ground truth testing and evaluating the accuracy of reconstructed meshes. Mesh statistics relating to the geometry of output surfaces is provided in Section 5.2. In Section 5.3 the proposed rule-based contour correspondence technique is evaluated through the use of visual inspection. This analysis is followed by a brief set of performance tests which assess implementation efficiency of the proposed method in Section 5.4. Section 5.5 presents a comparison of some of the properties of tested surface reconstruction methods, and how these properties suit specific applications. Experimentation is concluded with results of testing with a real data set in Section 5.6. Finally, in Section 5.7 implementation specific details of the proposed surface reconstruction algorithm is explained. Details for the hardware used in experimentation is also given in this section.

5.1 Reconstruction Accuracy

This section tests the similarity of reconstructed meshes with the ground truth meshes created in Section 4.2. Hausdorff distance was used to provide quantitative measurements for reconstruction accuracy. Hausdorff distance is a widely used metric used in the evaluation of both surface reconstruction and mesh simplification. In this section, Hausdorff distance is used to evaluate the similarity of ground truth

test models with reconstructions. The proposed surface reconstruction technique is compared with a set of free and open source implementations of existing surface reconstruction techniques.

Voronoi Filtering [24], Marching Cubes (RIMLS) [23], Marching Cubes (APSS) [22] and Screened Poisson surface reconstruction [40] are used to provide comparative analysis for the proposed surface reconstruction technique. These methods were chosen for comparison as they have free and open source implementations available in Meshlab [3], increasing the reproducibility of results. It is worth noting that all of these techniques are capable of some configuration. For the sake of testing, default configuration available in Meshlab is used for reconstruction with these techniques. We acknowledge that with optimal configuration there may be improvements in accuracy for these techniques. However, it is unreasonable for these parameters to be configured for each specific case in real applications, especially considering that real extracted data contains an assortment of complex structures. Use of default parameters also allows for more reproducible results. It is also worth noting that Marching Cubes (RIMLS), Marching Cubes (APSS) and Screened Poisson reconstruction techniques all require vertex normal information. Accurate vertex normal estimation is provided using Meshlab where possible for reconstruction techniques that require it.

Each mesh was tested with a range of sampled contours. In this chapter, *contour sampling rate* refers to the number of test slices generated per model. The *point sampling rate* refers to the number of points generated on a specific contour. In general, test contours are generated with similar sizes to contours observed in real data (between 10 to 200 point contours), although this was not held strictly to a particular value in any case.

Tests are performed with different contour sampling rates to see how each reconstruction method is affected by the varying resolution of source input. In real data, bifurcations result in smaller and smaller structures. These variable size structures are simulated in test contours with the ability to vary the number of test slices sampled from the ground truth models. The multiple branching model contains both relatively small and relatively large branching structures for similar reasons.

Hausdorff Distance:

Hausdorff distance provides a metric for the distance from a sample point to a mesh. In this analysis, it is used for comparing 3D surfaces. Hausdorff distance measurements reported by Meshlab include the minimum, maximum, mean and Root Mean Square (RMS). Hausdorff distance in this analysis requires a ground truth mesh to compare with reconstructed models. This requirement is the main reason test models were generated.

Recall that Hausdorff distance is an asymmetric metric. Hausdorff distance from the source model to the reconstructed model is useful for showing how well the reconstruction represents the original model. Hausdorff distance from the reconstructed model to the source model can be used to show how much extra information is included in the reconstructed mesh. For this reason, measurements are presented for Hausdorff distance in both directions.

Hausdorff distance measurements are all collected using Meshlab [3]. Hausdorff distance measurements are taken from the ground truth model to the reconstructed model, and also from the reconstructed model to the ground truth. The number of sample points taken for each model corresponds to the number of vertices in the source mesh. Source sample points are randomly created on vertices, faces, and edges of the source mesh. Mean Hausdorff distance measurements are presented while minimum, maximum measurements are omitted. Minimum and maximum measurements often do not provide much useful information about the distance between meshes aside from their most extreme differences. Average distance from sample points to the other mesh allows for more general analysis of mesh reconstructions.

All asymmetric Hausdorff distance measurements are normalised by the maximum average Hausdorff distance for that test mesh. Since all of these results are used for comparative analysis, their relative measurements are essential, and so normalised values provide clarity here. All Hausdorff distance measurements are rounded to 4 decimal places. In the case of erroneous results or unreportable measurements, a "-" is presented in the table. Explanations as to why results are unable to be reported in these specific cases are provided in the analysis where they occur.

5.1.1 Simple Model Test Set

The first test model is the simplest. The model is reconstructed from a test set, containing sampled points from the original model. Figure 5.1 shows the source mesh alongside reconstructions from the proposed method and Screened Poisson surface reconstruction.

In some lower levels of contour sampling, specific measurements are unavailable due to errors in reconstructed meshes. These errors were observed with Marching Cubes (RIMLS) where there were only 10 sampled slices per model. The implementation of this reconstruction technique consistently failed to produce a mesh with measurable mesh statistics in cases where 10 samples were given for any tested model.

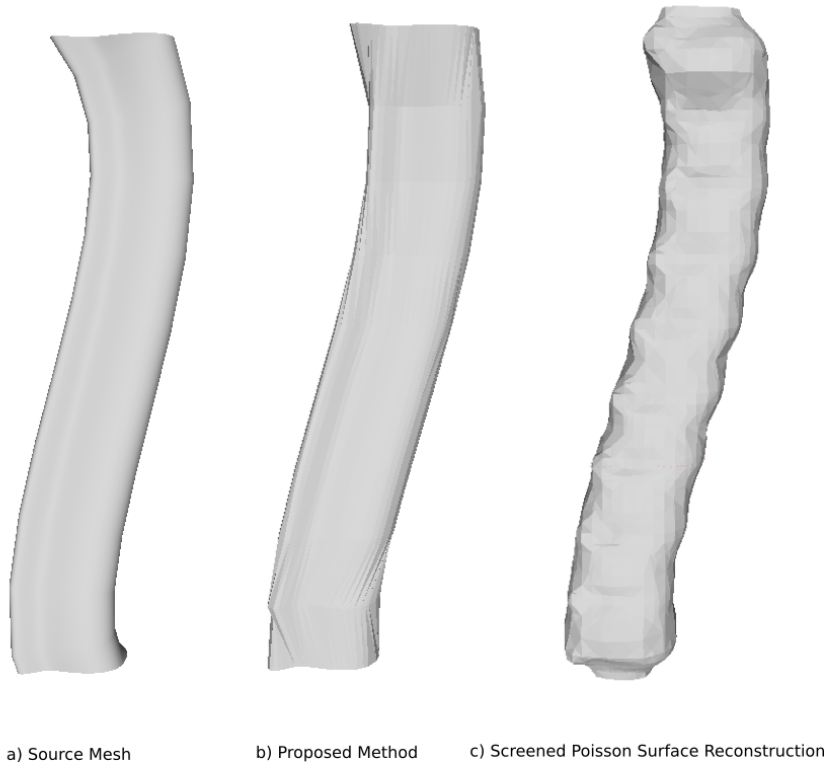


FIGURE 5.1: Reconstructions of a simple test model from 10 test slices.

Table 5.1 presents the mean asymmetric Hausdorff distance from the simple model to reconstructions of this model. The mean Hausdorff distance from reconstructed models to the original model is shown in Table 5.2. The trends for symmetric mean Hausdorff distance are shown in Figure 5.2.

	Number of Slices Sampled				
Method	10	20	30	40	50
Voronoi Filtering	1	0.2936	0.2134	0.2347	0.1961
Screened Poisson Surface Reconstruction	0.4688	0.3199	0.2157	0.2057	0.1559
Marching Cubes (APSS)	0.4277	0.3138	0.2437	0.1559	0.1308
Marching Cubes (RIMLS)	-	0.3007	0.162	0.139	0.119
Proposed Method	0.3323	0.2833	0.223	0.2453	0.1993

TABLE 5.1: Asymmetric mean Hausdorff distance from source to reconstructed mesh.

Table 5.1 shows that as the number of slices sampled increases, the mean Hausdorff distance decreases. That is, the distance between sample points from the source model to the reconstructed surfaces are on average smaller. The simplicity of this model allowed for quality reconstructions from all techniques when the contour sampling rate was sufficiently high, reflected by the Hausdorff distance measurements. Voronoi Filtering gives the highest average Hausdorff distance in the case where 10 sample slices are tested. Measurements converge to similar values where 20 and 30 sample slices are used for reconstruction. Screened Poisson, Marching Cubes (APSS) and Marching Cubes (RIMLS) all provide relatively low mean Hausdorff distance measurements for higher sampling rates. Marching Cubes (RIMLS) provided the lowest average Hausdorff distance for 30, 40 and 50 samples. The proposed technique showed low average Hausdorff distance measures when the contour sampling rate is low, though it had the highest average for both 40 and 50 samples.

	Number of Slices Sampled				
Method	10	20	30	40	50
Voronoi Filtering	0.3098	0.1427	0.0699	0.0664	0.0182
Screened Poisson Surface Reconstruction	0.7534	0.8361	0.2785	0.2318	0.1787
Marching Cubes (APSS)	1	0.7345	0.2613	0.1043	0.1119
Marching Cubes (RIMLS)	-	0.3143	0.1769	0.1115	0.1208
Proposed Method	0.0921	0.0318	0.0191	0.0138	0.0127

TABLE 5.2: Asymmetric mean Hausdorff distance from reconstruction to source mesh.

Screened Poisson and Marching Cubes (APSS) show relatively high average Hausdorff distance in Table 5.2 with lower sampling rates. For 10 and 20 sample slices these techniques showed a considerably higher average mesh distance to the source model in comparison to the proposed technique. These high average distances are

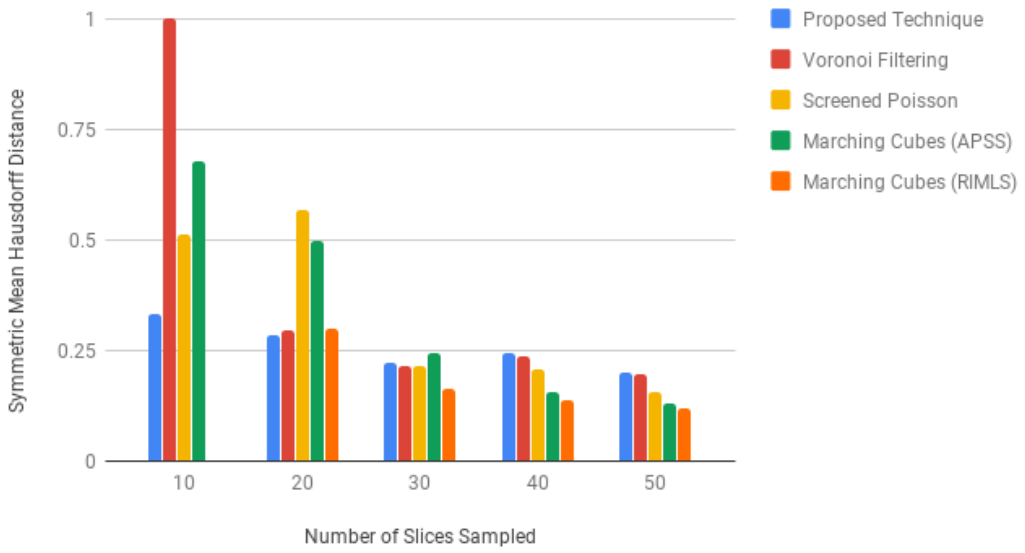


FIGURE 5.2: Symmetric mean Hausdorff distance for simple model.

likely due to the over-representation of source data with these techniques. Reconstructions using the proposed technique had considerably lower mean Hausdorff distance measurements than other techniques when samples were taken from reconstructions. The models reconstructed for this test data set with the proposed technique contained little extraneous information when compared to the original.

Both Marching Cubes (APSS) and Marching Cubes (RIMLS) created large meshes that had a considerable amount of issues in the meshes when the contour sampling rate was low. The surfaces produced from these methods is shown in Figure 5.3

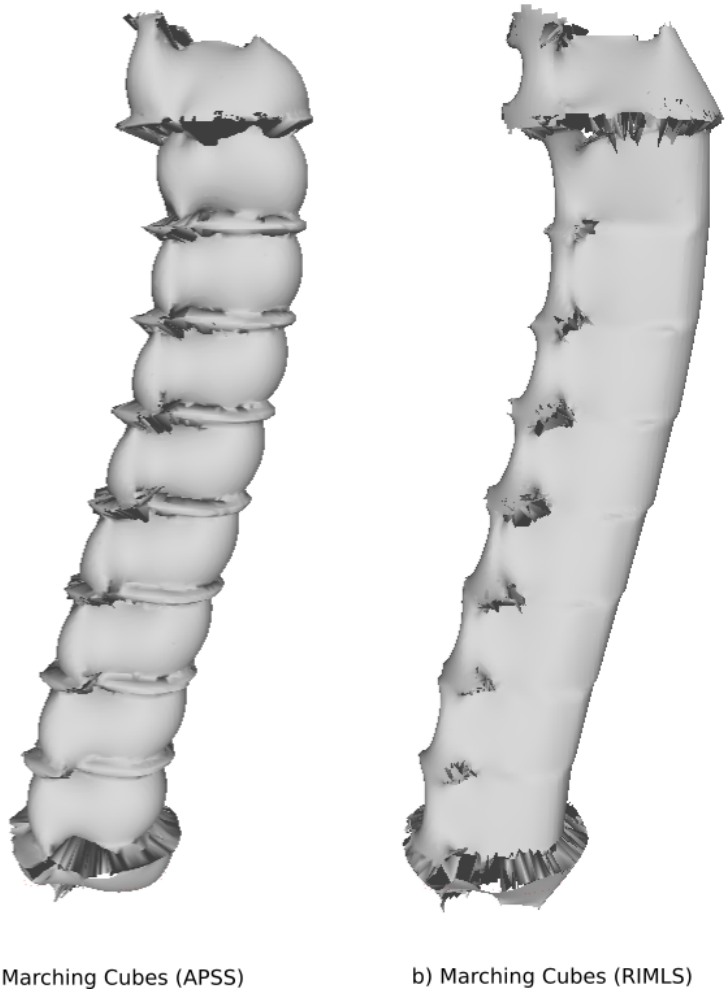


FIGURE 5.3: Issues in reconstruction from APSS and RIMLS with low contour sampling rates (10 slices).

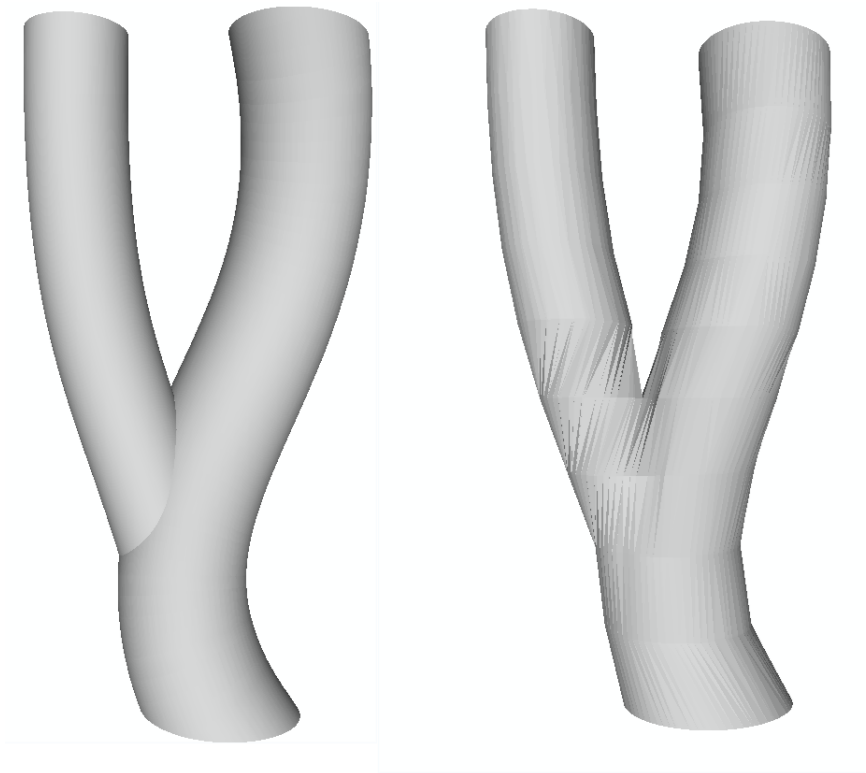
The proposed technique showed the lowest average Hausdorff distance from the reconstruction to the source mesh for all sample sizes of this model. For 10 sample slices, it provided less than 10% of the average distance of the maximum. Upon investigation of the reconstructed models, this was likely due to the reconstructions from other techniques containing extraneous information. Aside from the lowest tested contour sampling rate, Voronoi Filtering also had a relatively small average reconstruction to source Hausdorff distance. Both Voronoi Filtering and the DTW based correspondence use points directly in the reconstructed mesh. Since the other tested techniques provide approximate representations of sample data, they can tend to include extra information in reconstructions.

General trends of reconstructions for this test mesh are shown in Figure 5.2. This figure shows the symmetric Hausdorff distance, where the maximum of asymmetric

measures is taken (before normalisation). In general, this shows the resilience of the proposed technique towards low sampling rates. Though it tends to provide worse results for larger sample sizes in comparison to other techniques, the difference is relatively small.

5.1.2 Simple Branch Test Set

The branching model introduces a simple branch case for surface reconstruction techniques to handle. This model and the reconstruction from 10 samples using the proposed method are shown in Figure 5.4. The reconstruction in Figure 5.4b shows the triangulation generated in the branching case using the proposed technique. The importance of this model lies in the desired application of medical imaging. The ability to triangulate branching structures reflects well on a techniques ability to reconstruct complex structures such as branching airways in lungs. The reconstruction created by the proposed technique in Figure 5.4b shows a logical triangulation of a branching structure with only 10 contour samples.



a) Source Mesh b) Reconstructed Mesh

FIGURE 5.4: Test model compared with reconstruction from proposed technique (10 samples).

Mean Hausdorff distance statistics from the simple branching model to reconstructions are available in Table 5.3. In general, similar trends are shown with Hausdorff distance as with the first test model. The proposed technique again handles lower contour sampling rates well. However, when the sampling rate increases, both tested variants of Marching Cubes, and Screened Poisson surface reconstruction have smaller average Hausdorff distance. With a higher amount of sample points, Voronoi Filtering and the proposed technique share similar average Hausdorff distances. Screened Poisson surface reconstruction generated the least similar model to the ground truth where 10 slices were sampled. In contrast it had the smallest average distance from the reconstruction with 50 samples.

	Number of Slices Sampled				
Method	10	20	30	40	50
Voronoi Filtering	0.9987	0.1554	0.146	0.1138	0.1323
Screened Poisson Surface Reconstruction	1	0.0943	0.072	0.0649	0.0651
Marching Cubes (APSS)	0.3244	0.0866	0.0686	0.0645	0.0668
Marching Cubes (RIMLS)	-	0.1012	0.0786	0.0757	0.0744
Proposed Method	0.1994	0.1648	0.1526	0.123	0.1385

TABLE 5.3: Mean Hausdorff distance from simple branch ground truth to reconstructed model.

	Number of Slices Sampled				
Method	10	20	30	40	50
Voronoi Filtering	0.2799	0.1197	0.0796	0.046	0.0342
Screened Poisson Surface Reconstruction	0.4923	0.127	0.1172	0.1866	0.0967
Marching Cubes (APSS)	1	0.1258	0.0658	0.0888	0.047
Marching Cubes (RIMLS)	-	0.1892	0.108	0.1255	0.0714
Proposed Method	0.1397	0.0583	0.0566	0.2373	0.0755

TABLE 5.4: Mean Hausdorff distance from reconstruction to simple branch ground truth.

Marching Cubes (APSS) provided the greatest over-representation of the original ground truth model with 10 samples, as can be seen in Table 5.3. Voronoi Filtering

provided relatively low average Hausdorff distance from the reconstruction to the ground truth mesh for higher sampling rates of this test model. The proposed technique generally provides low distance measurements from the reconstructed model to the ground truth. An exception is the case where 40 sample slices are given.

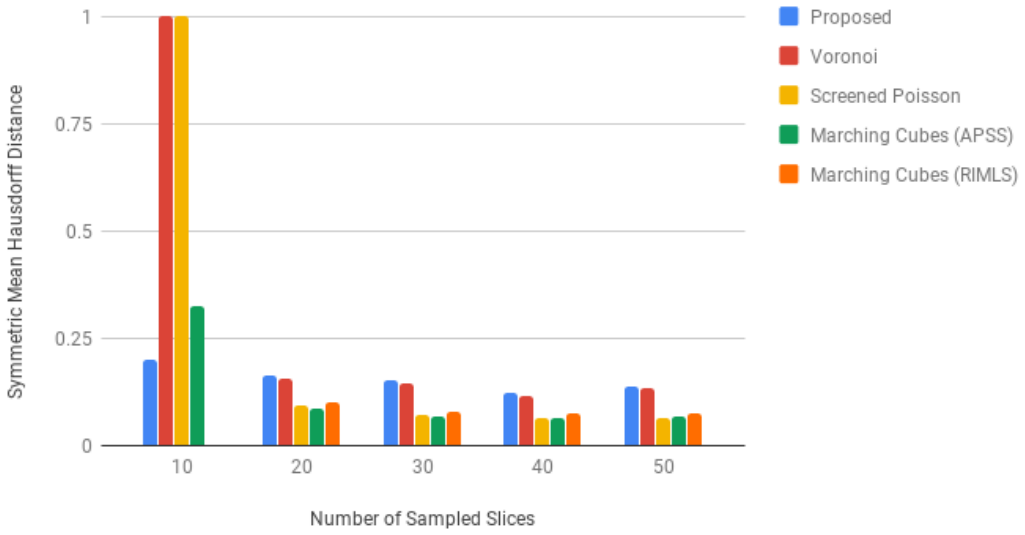


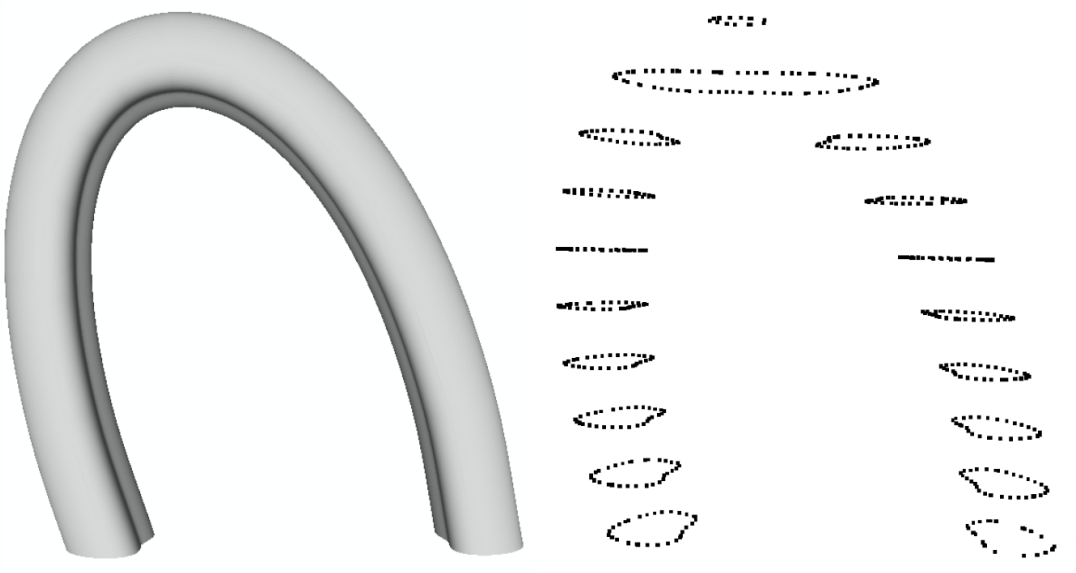
FIGURE 5.5: Symmetric mean Hausdorff distance for simple branching mesh.

Figure 5.5 shows trends for symmetric mean Hausdorff distance. Similar to the first mesh, the proposed technique shows resilience to low contour sampling rates. Measurements for this test mesh showed a more clear difference between techniques in methods with more contour samples available. The test model itself is represented by a smooth branch, with minimal deformations. The likely reason for Screened Poisson, Marching Cubes (APSS), and Marching Cubes (RIMLS) providing relatively low average Hausdorff distance, in this case, is their ability to represent smooth surfaces.

5.1.3 Bent Branch Test Set

The next tested branching case is the bent tube case, shown in Figure 5.6 alongside a test configuration. The main difference between this problem and a branch caused by bifurcation is that the difference in size of contours is much less predictable. The size variance between the branching contours in this mesh is dependent on the shape

of the bend in the tube. Also, the sampled contours rapidly get smaller near the top of the mesh.



a) Bent Tube Mesh b) Sample Points

FIGURE 5.6: Bent tube with an example of generated test data (10 slices)

Mean Hausdorff distance measurements are given in Table 5.5. The proposed technique has the lowest mean Hausdorff distance when given only 10 sampled slices. The accuracy of the other techniques is relatively similar when the contour sampling rate is increased. The proposed technique, however, has the highest mean Hausdorff distance for sampling rates of 40 and 50.

	Number of Slices Sampled				
Method	10	20	30	40	50
Voronoi Filtering	1	0.152	0.1072	0.0658	0.0659
Screened Poisson Surface Reconstruction	0.9337	0.2774	0.1571	0.1361	0.1248
Marching Cubes (APSS)	0.6791	0.2028	0.1268	0.1071	0.0948
Marching Cubes (RIMLS)	-	0.5171	0.1536	0.1278	0.1103
Proposed Method	0.2982	0.1513	0.1068	0.1103	0.0864

TABLE 5.5: Mean Hausdorff distance from source to reconstructed model for bent tube test mesh.

	Number of Slices Sampled				
Method	10	20	30	40	50
Voronoi Filtering	0.1642	0.0506	0.0372	0.0251	0.0229
Screened Poisson Surface Reconstruction	0.8505	0.1618	0.0752	0.0832	0.0789
Marching Cubes (APSS)	1	0.2498	0.0741	0.0609	0.0516
Marching Cubes (RIMLS)	-	0.139	0.1102	0.0808	0.0672
Proposed Method	0.1298	0.0618	0.0286	0.1398	0.0964

TABLE 5.6: Mean Hausdorff distance from reconstruction to source model for bent tube test mesh.

The mean Hausdorff distance measurements from the reconstruction to the source model in Table 5.6 is where some differences start to show when compared to the first two test models. Although the proposed contour-based surface reconstruction technique shows the lowest mean distance to the ground truth with 10 samples, it provides the highest measurements for higher sampling rates. This technique performed well in creating a branching triangulation for the bifurcating structure. However, the size difference of contours in this branching case caused issues in most reconstruction methods. The most significant difference in the reconstructions of this model was observed to occur at the branching contours, as expected.

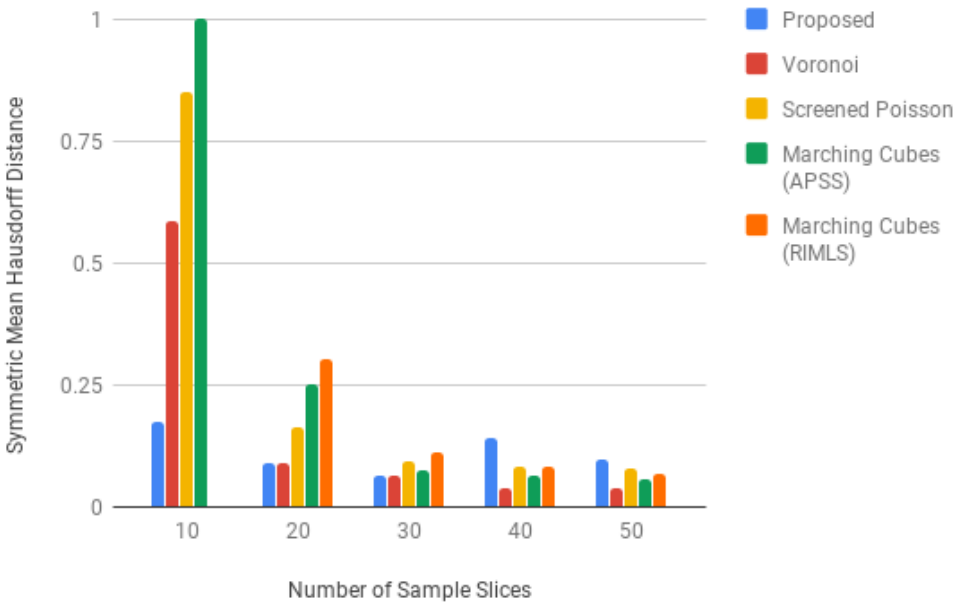


FIGURE 5.7: Symmetric mean Hausdorff distance for bent model.

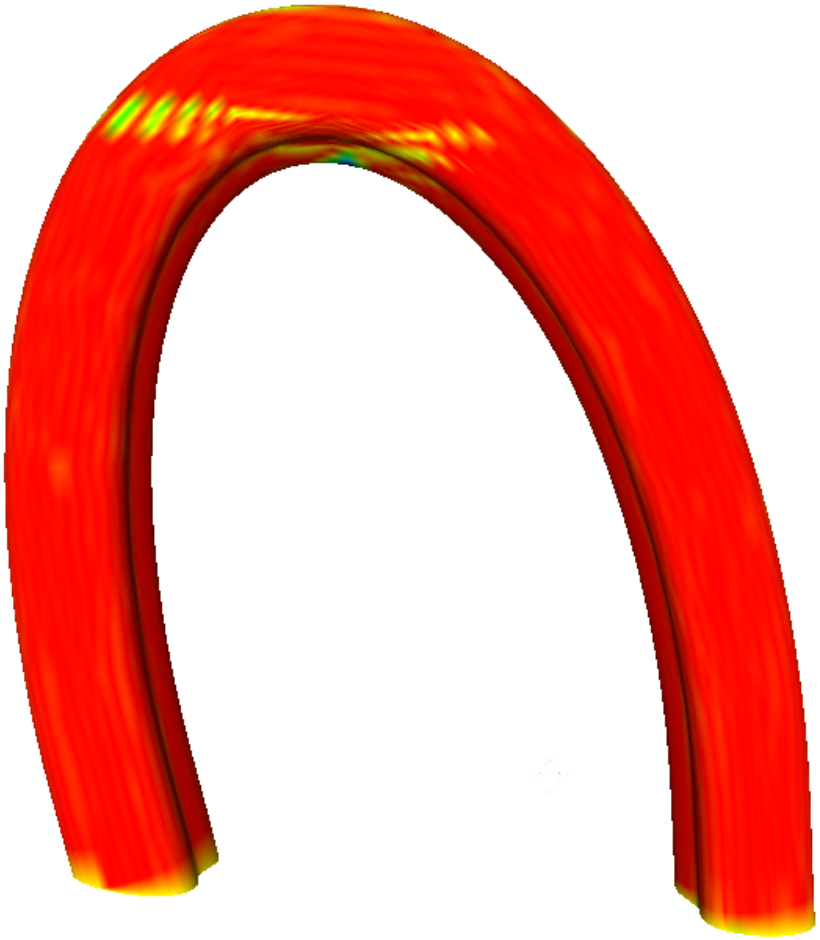


FIGURE 5.8: Heat map for mean Hausdorff distance for bent tube model (40 samples). Reconstructed with the proposed method. Red is less distance, blue is more.

Figure 5.7 shows the general trends of measurements. The proposed method again performs well with low sampling rates. Voronoi Filtering surprisingly has the lowest average symmetric distance for 40 and 50 samples. The proposed technique has relatively high measurements for both of these tests. Notably, these measurements are considerably higher than the cases where 20 and 30 slices are sampled. In the heat map in Figure 5.8, the largest distances in the mesh occur near the case where two contours are matched to one. This difference is likely due to an observed "twisting" effect that appears to happen often in cases where alignments are attempted between a long feature sequence and a much smaller one. The use of an adjustment window constraint can be considered for preventing the effects of such alignments. Use of a global constraint such as the Sakoe-Chiba constraint [53]

can prevent the warping path from straying too far from the diagonal of the DTW matrix. This constraint stops excessively long sequences being matched to shorter ones.

5.1.4 Multiple Branching Test Set

A more complex test data set is provided by sampling the multiple branching test mesh. A rendering of this mesh is provided in Figure 5.10. This mesh consists of two large branches in either direction. One side also contains further smaller branching subdivisions. Mean Hausdorff distance measurements are given in Table 5.7 and Table 5.8. The difference in size of branching contours highlights the resilience of reconstruction techniques.

	Number of Slices Sampled				
Method	10	20	30	40	50
Voronoi Filtering	1	0.289	0.0207	0.0651	0.0328
Screened Poisson Surface Reconstruction	-	-	0.091	0.0604	0.0472
Marching Cubes (APSS)	-	-	0.0597	0.0476	0.0334
Marching Cubes (RIMLS)	-	-	0.071	0.0541	0.0385
Proposed Method	0.8284	0.1532	0.0571	0.0556	0.0337

TABLE 5.7: Mean Hausdorff distance from source to reconstruction for multiple branching test model.

	Number of Slices Sampled				
Method	10	20	30	40	50
Voronoi Filtering	1	0.2252	0.017	0.0897	0.0662
Screened Poisson Surface Reconstruction	-	-	0.4216	0.1665	0.1652
Marching Cubes (APSS)	-	-	0.4642	0.1697	0.1543
Marching Cubes (RIMLS)	-	-	0.5195	0.2398	0.1999
Proposed Method	0.657	0.2943	0.2155	0.2393	0.1561

TABLE 5.8: Mean Hausdorff distance from source to reconstruction for multiple branching test model.

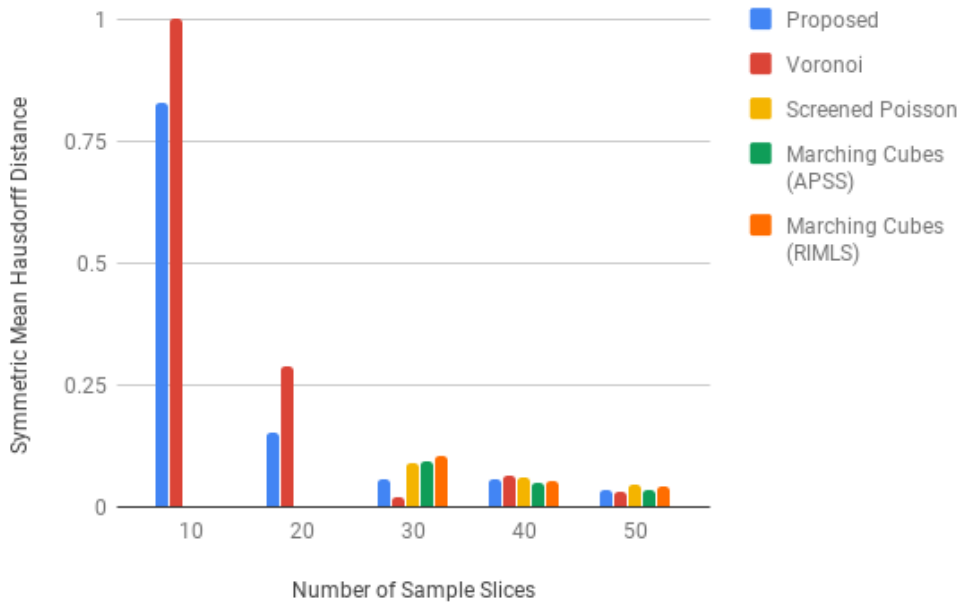


FIGURE 5.9: Symmetric mean Hausdorff distance for multiple branching model.

In general, similar distance values are observed for the higher sampling rates of all reconstruction techniques. For lower sampling rates accurate estimates for point normal information was unable to be obtained. Meshes are not reconstructed from this inaccurate point normal information as it unfairly reflects poorly on the reconstruction technique. Comparisons are still available with Voronoi Filtering, but it does emphasise an issue with reconstruction techniques that rely on normal information. In more straightforward cases, vertex normal information can be correctly estimated, but when more complex, under-sampled data sets are provided this is not the case. With the desired application in medical imaging, it is expected that complex structures will occur, so the ability to obtain accurate vertex normal information is important.

With both Voronoi Filtering and the DTW based technique there is a considerable amount of reconstruction errors with the lower contour sampling rates. These errors are reflected by the symmetric mean Hausdorff distance trends shown in Figure 5.9. The proposed method has a notably smaller mean Hausdorff distance than the reconstruction from Voronoi Filtering with 10 and 20 samples. Another notable result is the comparably small result for symmetric mean Hausdorff distance with Voronoi Filtering specifically in the case with 30 sample slices. A reconstruction

using the DTW point correspondence method is given in Figure 5.10b. With the proposed technique larger branches can be successfully reconstructed with no obvious errors. However, there are noted issues with contour correspondence with the smaller branches. This model is the first test model for which the proposed technique shows issues in contour correspondence. The proposed method was able to correctly correspond contours for all tested input data set sizes for all other models. However, for this model, there are branching structures whose centroids appear to stay relatively close after the branch. The implemented contour correspondence technique fails to establish one-to-one correspondences for contours following this branch.

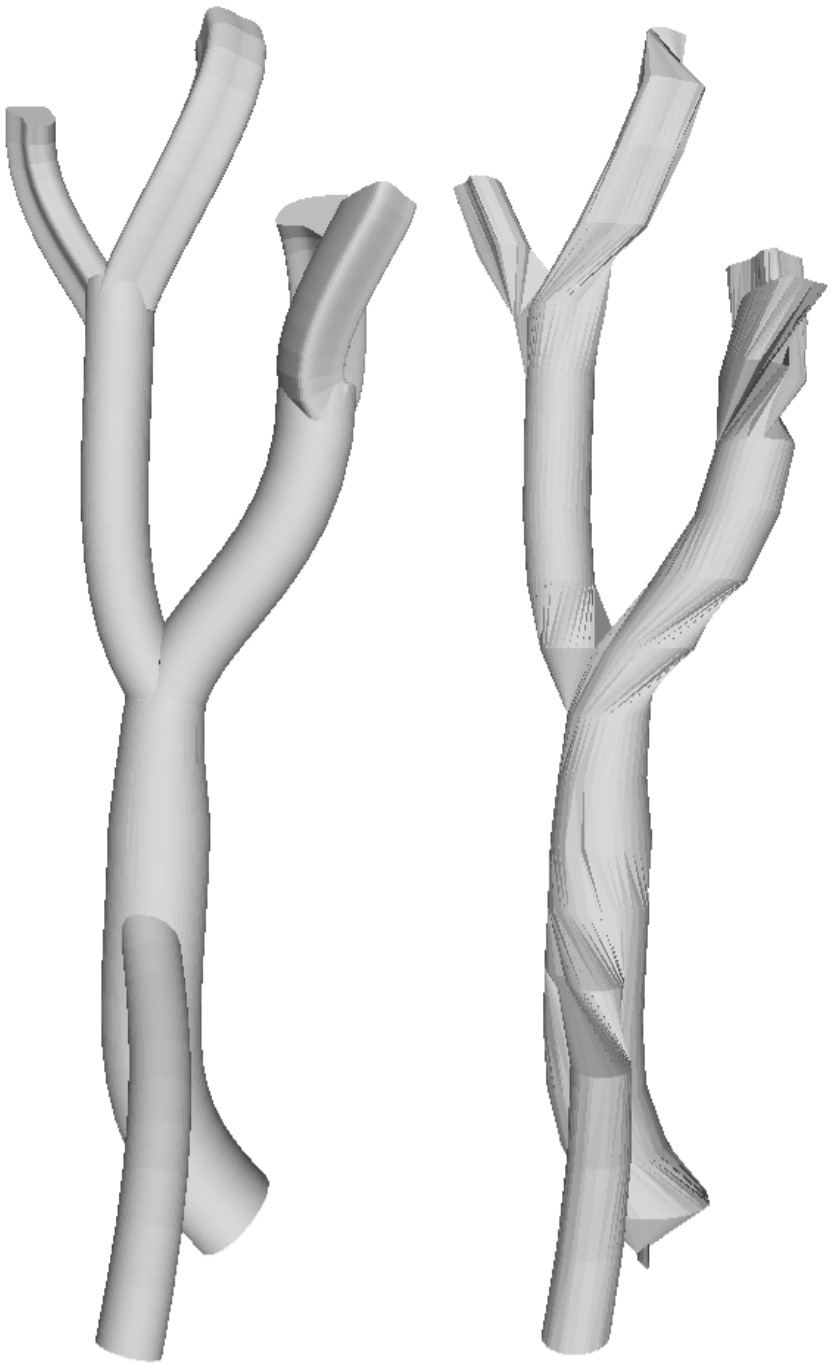


FIGURE 5.10: Multiple branching structure and a reconstruction from 20 test slices.

Figure 5.10b shows conical shapes generated near the branching contours in this test set. In the DTW warping path, these features correspond to a long series of points being matched to very few points. This case seems to happen more often when smaller contours are matched to considerably larger ones, which tends to be

the case in branching contours. Slope constraints or other global warping path constraints may be used to avoid such long sequences of points being matched to shorter ones.

5.1.5 Summary

The set of ground truth testing models provides a useful preliminary check for accuracy in the reconstruction of specific cases expected in medical imaging. Hausdorff distance measurements showed that the proposed technique was resilient with low sample rates, consistently providing the most similar meshes with only 10 samples per test model. The method also consistently showed relatively low results for reconstruction to source Hausdorff distance measurements, meaning that it did not tend to over-represent the mesh as much as other techniques did. Visual inspection of branching models shows that logical triangulations are generally created successfully. However, there is noted issues with a visually displeasing twisting effect that can occur in certain situations.

The comparative analysis also shows that the proposed contour-based reconstruction method has comparable accuracy to the point-cloud surface reconstruction techniques in higher sampling rates. When there is a sufficient number of samples, fitting techniques such as APSS and RIMLS work well, as does Screened Poisson surface reconstruction. APSS, RIMLS and Screened Poisson all produce implicit surfaces. Since all the test models are relatively smooth surfaces, implicit surfaces likely represent these smooth surfaces well. The property of implicit surfaces that allows for better representation of smoother surfaces is the likely reason why these techniques tend to slightly outperform the proposed technique and Voronoi Filtering at higher sampling rates. However, the requirement for vertex normal information reduces the ability to reconstruct problems which do not have enough samples to provide an accurate vertex normal estimation. Measurements for Voronoi Filtering show that in general it can work well with a large number of sample points, but has high average Hausdorff distances where the surface is under-sampled. Observations of reconstructions show that this technique can omit information when under-sampling occurs. These observations reflect what has been inferred from similar observations in literature [24].

Further comparison with more surface reconstruction techniques remains as future work. This section only provides a comparison with point-cloud surface reconstruction techniques with free implementations available in Meshlab. Data gathering which includes more techniques, such as other contour-based surface reconstruction techniques, would provide a greater understanding of suitability for applications in medical imaging.

5.2 Mesh Statistics

This section presents various size statistics about reconstructed meshes. Notable information about reconstructed meshes in this section includes the output mesh size (face count), the total length of all edges in a reconstructed mesh, and surface area information. The number of faces is included as it gives a comparison of the output mesh sizes to expect with each method. Smaller output meshes are preferable for this application as the source data sets in real scenarios contain a large amount of information. Surface simplification may be used for methods that produce larger meshes, but it may reduce the accuracy of the reconstructed mesh further. All mesh size statistics are computed using Meshlab [3]. Total surface area is also an interesting measurement as it shows how the surface area of reconstructions varies with different sampling rates.

5.2.1 Output Mesh Size

Because the problem of surface reconstruction in medical imaging can contain a large number of sample points, it is essential to consider the size of the output mesh. It may be unsuitable for rendering in specific applications if a method produces considerably large meshes. For this reason, the size of the output meshes generated by the tested reconstruction methods is measured. Table 5.9 shows the number of faces in generated reconstructions of the simple test model. Note that while Screened Poisson, APSS and RIMLS produce implicit surfaces, piecewise-planar meshes are produced from surfaces in implementation in Meshlab. Meshlab default settings are used for all reconstructions.

	Number of Slices Sampled				
	10	20	30	40	50
Voronoi Filtering	5093	8924	1127	13476	15661
Screened Poisson Surface Reconstruction	9324	10908	10916	11640	27234
Marching Cubes (APSS)	108141	89693	88062	85793	86240
Marching Cubes (RIMLS)	94955	90870	88685	86386	86566
Proposed Method	1894	4032	6814	8333	10519

TABLE 5.9: Number of faces produced by each reconstruction method using samples from the simple test model.

The size of the output mesh for Voronoi filtering and the proposed technique tend to scale with the input data set size. This is mostly because these techniques use the sample points directly as part of the reconstructed mesh. Since the other techniques use approximations of the sample points to generate an implicit surface, they are not as predictable in terms of output mesh size. These two techniques generally provide fewer mesh faces than the other tested methods. The proposed method notably reconstructed only 1894 faces for the case where 10 test slices are given.

Screened Poisson produces a mesh with much fewer faces than the marching cubes techniques despite having comparable reconstruction accuracy. Both of the tested Marching Cubes techniques produced a considerably large amount of faces. Interestingly, the largest meshes produced for these techniques were generated where there was a lower contour sampling rate. Meshes produced for these cases produce a considerable amount of issues, shown in Figure 5.3. These reconstruction issues are likely the cause of the relatively large mesh sizes.

5.2.2 Mesh Geometry

The following sets of measurements are for reconstructions of the simple branching ground-truth mesh. Total mesh area for reconstructed meshes is given in Table 5.10. Perhaps the most notable feature about this set of data is that surface area of reconstructions converges to a similar value as the contour sampling rate increases. With more sample data available, techniques provide a tighter bound on the ground truth

surfaces. Aside from Voronoi Filtering, all reconstruction techniques have a very similar surface area in cases where 30, 40, and 50 slices are sampled from the test mesh. These measurements show that the meshes generated from Voronoi Filtering are generally inconsistent sizes, while other techniques provide relatively constant total mesh area. Another notable feature is that the proposed method generally tends to underestimate the surface, and grows larger when more sample data is available. The other techniques all overestimate and then converge down to a similar value.

	Number of Slices Sampled				
Method	10	20	30	40	50
Voronoi Filtering	364.17	304.06	249.75	228.04	208.69
Screened Poisson Surface Reconstruction	150.85	144.6	145.16	148.25	146.45
Marching Cubes (APSS)	211.63	146.74	144.06	145.92	143.09
Marching Cubes (RIMLS)	-	148.51	145.58	146.35	143.57
Proposed Method	137.95	137.5	143.28	166.32	144.35

TABLE 5.10: Surface area for whole reconstructed branching mesh. Rounded to 2 decimal places.

Table 5.11 gives the total edge length for meshes reconstructed from the simple branch test set. The cost function used in DTW for point alignment was Euclidean distance between points on contours. Since this aims to minimise the total Euclidean distance of these edges a relatively small value for the proposed method is expected here. The total edge length is lowest for all tested sample sizes in the proposed method, aside from one case where Screened Poisson provides the lowest total edge length. The total edge length understandably increases as more source points are included as this technique includes all source points in the final model. Another notable feature in this data is that meshes reconstructed from Marching Cubes using the APSS and RIMLS fitting techniques had smaller total edge lengths where more source data is available.

	Number of Slices Sampled				
Method	10	20	30	40	50
Voronoi Filtering	4783.92	4321.37	3981.99	3954.99	3925.21
Screened Poisson Surface Reconstruction	2865.55	3075.21	3085.21	3180.74	4701.54
Marching Cubes (APSS)	12270.03	9012.7	8846.64	8779.18	8705.41
Marching Cubes (RIMLS)	-	9110.22	8906.04	8816.62	8740.46
Proposed Method	2617.69	2766.07	2929.54	3266.46	3343.49

TABLE 5.11: Total edge length for reconstructed models.

5.2.3 Summary

Surface area statistics gives insight on how the size of reconstructions varies depending on sample sizes. For the ground truth mesh tested, Voronoi Filtering and the proposed technique generally provide smaller output mesh sizes. Visual analysis of produced meshes leads to the conclusion that this is likely due to these two techniques under-representing source data. The other tested techniques tend to include extra information. The proposed technique output meshes vary around 20% at most for different input sample sizes.

5.3 Contour Correspondence Evaluation

This section provides a brief evaluation of the proposed rule-based contour correspondence method used in the reconstruction of surfaces. Since it is visible when contour correspondence is incorrectly computed, a different set of test data is used to the ground truth method. The method described in Section 4.4 is used to quickly generate a test data set which provides a considerable amount of contour correspondence challenges. The source image, shown in Figure 4.7, is used to generate 83 slices, with a spacing of 10 rows between each slice.

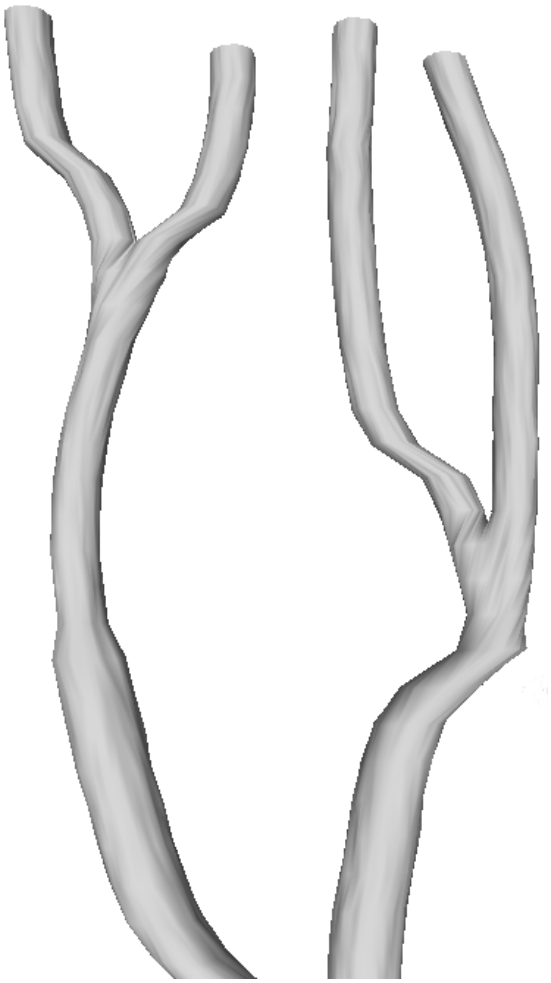
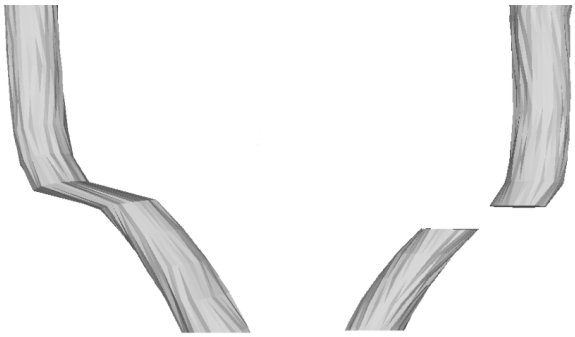


FIGURE 5.11: Reconstructed branching model.

Where contour correspondence is successful, point correspondence using DTW generally creates a triangulation which visually represents the branching structure well. Figure 5.11 shows the result of successfully reconstructed branching structure using the proposed method. Each branching structure has correct correspondence, and the triangulation generated represents this information well.



a) Match (high threshold) b) Failed match

FIGURE 5.12: Matching distant contours in contour correspondence.

Figure 5.12b shows the result of contour correspondence failing to match contours because of the distance between contour centroids being too large with the current threshold. Increasing this threshold will fix this particular issue (Figure 5.12a), but it generates other issues with matching too many contours elsewhere in reconstruction. This issue is observed to occur in cases where tubular structures bend along the plane which the contours are sampled.

a) Correct correspondence b) Incorrect correspondence

FIGURE 5.13: Matching too many contours in correspondence.

In some cases, the rule-based correspondence technique matches too many contours. This particular issue seems to happen when bifurcating structures remain near enough to each other after a bifurcation happens. Figure 5.13a shows the result triangulation from a correct contour correspondence, while Figure 5.13b shows the result of matching too many contours. This erroneous correspondence happens because the contour centroids are still considered to be within the threshold range of each other after the branch has already occurred.

5.3.1 Summary

In its current state, the rule-based correspondence method matches contours correctly in most cases. With most branching cases, correspondence is correctly achieved, and quality triangulations are created. However, there are noted issues with specific contour configurations. Rule-based correspondence sufficient as a proof of concept but does not yet accurately correspond contours in a manner sufficient for applications with real data.

5.4 Reconstruction Performance

This section provides a brief analysis of implementation performance for the proposed surface reconstruction method. A brief evaluation of how space requirements grow with the input contour sizes is given. To measure running time in implementation reported CPU running time is measured for varied contour sizes. All timing measurements are performed on the same PC running the same OS. Since space requirements are expected to grow with contour sizes, RAM usage was observed while running tests to ensure that this was not exceeded. For all presented running times in this section, the average (mean) is given for five test runs. The proposed surface reconstruction algorithm was compiled with the GCC 8 compiler with default optimisations enabled. The targeted C++ standard is C++17.

Point correspondence

With the naive implementation of DTW in the proposed method, there is two observation series of length n and m. These observation series correspond to points of matched contours on different slices. The space requirements for the generation of this matrix is $O(nm)$. Traversing the generated warping path takes at least $max(n, m)$ steps for as many diagonal moves as possible and at most $n + m$ steps for no diagonal moves.

Contour correspondence

Expensive operations in the proposed rule-based contour correspondence technique include the computation of contour centroids and the computation of the farthest

point from the centroid. Both were implemented in a way that required n operations for a contour of size n. Finding the closest pair of points for two contours was implemented naively by comparing every point in a contour to every point in another contour. This process is used in branching contours only so occurs relatively infrequently. Problems such as finding the closest pair of points in contour correspondence may be optimised with ease.

5.4.1 Implementation Performance

Observations of real data provided show that, in this application, contours are expected to range mostly between the sizes of ten to two hundred points. To test the implementation performance, the number of points per contour are varied. The idea of this is to individually test how the performance of point correspondence and contour correspondence is affected.

By varying the number of points per contour, points can be randomly generated for a contour. This allows choice of the number of points to generate in a contour. For testing performance in point correspondence, an alignment is generated between two contours with the following size ranges: 10, 100, 1000, and 1000. Although contours larger than 1000 were not found in the provided real data set, higher values are included to see how larger contour sizes affect performance.

Contour Size	Contour Correspondence Time (ms)	Point Correspondence Time (ms)
10	0.029	0.051
100	0.023	1.663
1000	0.023	120.867
10000	0.187	11797.6

TABLE 5.12: Time taken in milliseconds for reconstruction of two adjacent contours of varied size.

Measurements in Table 5.12 show that the length of feature sequences influences the computation time for the DTW array and warping path. The matching of contours of size 10 results in a $10 \times 10 = 100$ size DTW array, and matching contours of size 10000 results in a size $10000 \times 10000 = 100000000$ size array. For contours of

size 10, the time taken is negligible. To reconstruct a problem with 1000 contours (assuming 1 per slice) this would take around 51ms for the point correspondence stage only, assuming a non-parallel implementation. However, issues arise when contour sizes start to increase past 1000. Consider a problem with 1000 contours of this size again; this would take around 12 seconds for only the point correspondence stage to complete (again assuming no parallel implementation). The real data investigated does not contain contours this large, but it is worth investigating how to handle this case should it arise with higher resolution modalities.

Contour correspondence time is not affected as much as the point correspondence stage by varying the size of the contour. It is only with an increase of contour size to 10000 that there is a notable change in time taken for contour correspondence, though this running time is still relatively small. The size and centroid computations performed in contour correspondence each require n operations for a contour of size n.

Contours Per Slice	Contour Correspondence Running Time (ms)
1	0.0237
10	0.9698
100	93.7686
1000	9119.502

TABLE 5.13: Time taken in milliseconds for reconstruction of two adjacent slices with the number of contours varied.

Contour correspondence time is expected to vary with the number of contours per slice. Table 5.13 shows the result of performing contour correspondence between two slices. Contour sizes are held at 50 points per contour. Points are randomly generated at distances at most 100 from the origin in any direction. This data is random, but it still shows the performance trends of implementation. It is reasonable to expect that contours per slice can reach around 100 often. In this case, 93.7686ms is not excessive running time. However, when the number of contours per slice approaches 1000 running time becomes unreasonable.

5.4.2 Summary

Time taken for point correspondence is observed to be strongly influenced by the size of input contours. While the contour correspondence is not affected by the size of contours as much, it does take considerably longer to generate correspondence for more contours on each slice.

The current implementation of point correspondence is simple and has room for improvement with efficiency in DTW. The current implementation of DTW also uses no global constraints. The use of an adjustment window constraint may be beneficial to both performance [51], and to prevent the matching of long feature sequences to excessively short ones. The naive contour correspondence implementation leaves room for efficiency improvement as well. Outside of these improvements, it is worth stating that this problem is parallelisable.

5.5 Comparison of Surface Reconstruction Algorithm Properties

This section discusses the properties of tested surface reconstruction techniques. How these properties affect their suitability to applications in medical imaging is described.

An advantage of the proposed technique for applications in medical imaging lies in the fact that every real data sample is used in the reconstructed surface. Techniques like Screened Poisson surface reconstruction are not be guaranteed to use all data points, or they may only provide an approximate fit to these data points. The reason for approximation is likely due to their intended application in point cloud surface reconstruction, where noise is expected to be present. Using an approximation can smooth out noisy data if required. Noise removal is performed during extraction of contours in medical imaging, so this expectation of noisy data can be relaxed. Similar to the proposed technique, Voronoi Filtering uses the sample points as part of the reconstructed mesh. However, it does not always use all sample points in reconstruction.

The proposed method does expect extracted contours to be in a specific format. One such expectation of this format is that points must be in a cyclic order in either clockwise or contour-clockwise direction. A mixture of both (or no ordering at all) will result in poor alignment of contour vertices and so in turn, a poorly reconstructed mesh. However, unlike most point-cloud reconstruction techniques normal information is not required. Voronoi Filtering also does not require normal information for reconstruction.

The most appropriate surface reconstruction technique may depend on the application. For example, some applications may not require that points are represented precisely in the final mesh. For such an application, point cloud reconstruction techniques that approximate source points may be more suitable. Brief summaries of reconstruction method properties are listed below.

- Voronoi Filtering uses source points in a reconstructed mesh, though it is not guaranteed to use all points. Vertex normal information is not required.
- Screened Poisson surface reconstruction provides an approximation of the source point set, requires vertex normal information and generates hole-free meshes.
- Similarly, Marching Cubes (APSS) and Marching Cubes (RIMLS) implementations in Meshlab [3] require vertex normal information and provide approximations of source points.
- The proposed contour-based reconstruction technique which implements DTW uses all source points in reconstruction and does not require vertex normal information.

5.5.1 Reproducibility

The importance of reproducibility in surface reconstruction techniques has been mentioned by Berger, Tagliasacchi, Seversky, *et al.* [11]. To gain reproducible results, open-source implementations of surface reconstruction techniques are useful. The open source implementations available in Meshlab [3] allow for great reproducibility of results. Techniques that do not provide open implementations can

lead to poor reproducibility of results as implementation takes time, and can result in misunderstanding or errors. For reproduction of results, test contour data is made available [1]. Source code for the proposed surface reconstruction technique is also made available [2]. Alternatively, repository access can be requested by emailing david.j.mackay411@gmail.com.

5.6 Reconstruction of Real Data

This section covers the reconstruction of contours extracted from a real data set. One of the main goals of this thesis was to investigate the application of DTW in contour-based surface reconstruction for medical imaging. Demonstrable application of the proposed technique with real data is therefore essential.

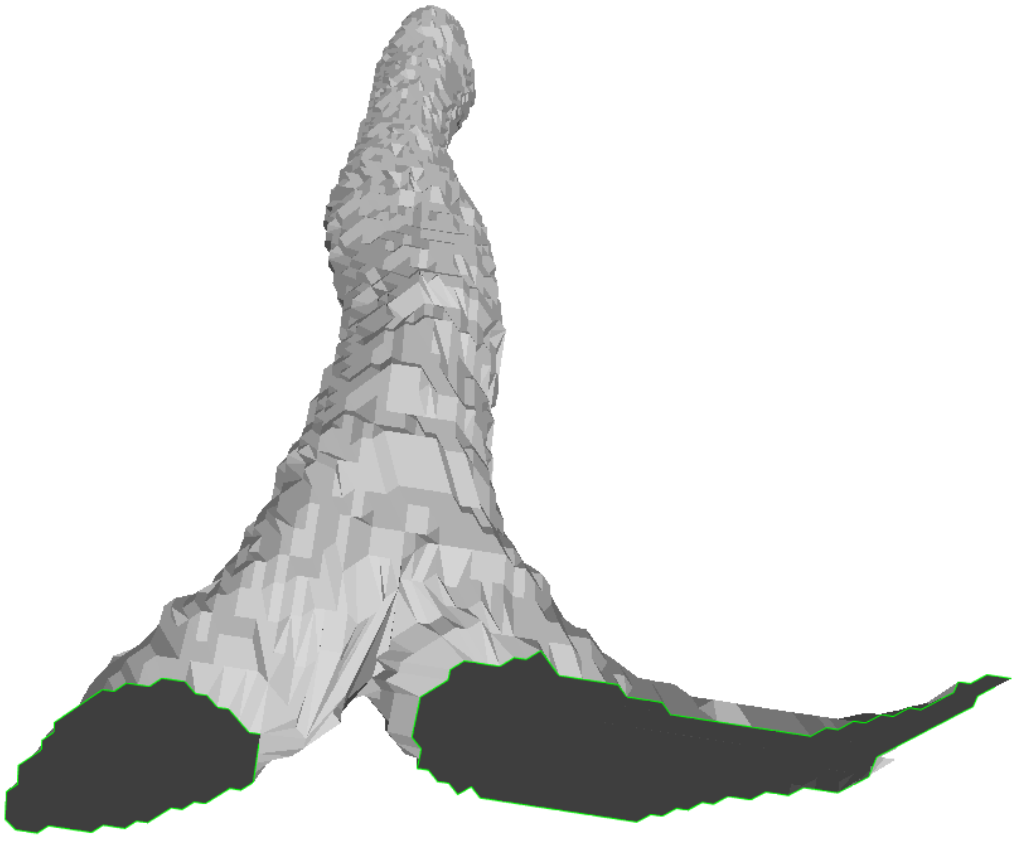


FIGURE 5.14: Reconstruction using a subset of real contour data extracted from HRCT slices.

[1] https://drive.google.com/open?id=1QYsQa-qsz9mwDRz_nk4ohtD4pITV9AD3
[2] https://drive.google.com/open?id=1EHIx3dOOwcP1KTxWxhv8NWyROHAyVZ-P

One of the most significant issues with contour-based surface reconstruction in medical imaging is the branching problem. Figure 5.14 shows the result of reconstructing extracted contours from a real data set. This data set includes a single division of the trachea into the two main bronchi. Smooth shading was deliberately not used to show the underlying polygonal structure of this reconstruction. Inspection of the figure shows the ability for the technique to produce a consistent point alignment and triangulation. The use of the contour-merging strategy for obtaining a triangulation between branching contours using DTW is also visible in this figure.

5.7 Implementation Details

All experimentation was performed on a desktop PC with the following hardware details:

- Operating System: Fedora Linux 29.
- CPU: Intel i5-4670K Quad Core @ 3.8GHz
- GPU: AMD Radeon R9 380x. 4GB GDDR5 Memory.
- RAM: 8GB DDR4.

5.7.1 Surface Reconstruction

Surface reconstruction algorithms were implemented using modern C++, making use of features like move semantics and object-oriented programming for performance and code re-use. Additional libraries used include Catch for unit testing, OpenGL Mathematics (GLM) [86] for graphics related math, Boost [87] for string parsing and file processing and CMake for project management.

5.7.2 Test Generation

Test models were generated with the use of Blender [83]. Blender3D is well documented and as a result of this is fast to learn. It was chosen because of its ease of use and permissive license. Blender3D allows the creation of high-quality meshes and a wide variety of formats. Test data is generated using the models and method

discussed in Chapter 4. Implementation of the contour-sampling tool was written in modern C++ using GLM [86] for mathematics, and Boost [87].

Chapter 6

Discussion and Conclusion

This thesis has carried out a detailed analysis of the contour and point correspondence problems commonly encountered in mesh reconstruction algorithms designed specifically for medical data. Solutions for branching structures were investigated and novel techniques introduced for handling these cases. The use of DTW for aligning points between contours allows for quality triangulations. A rule-based contour correspondence technique was also introduced. Combined with the contour merging technique, DTW can find alignments between branching contours that are consistent with structures expected in medical imaging. Generation of ground truth test models representing structures in medical imaging known to cause issues, combined with the proposed contour generation method, allows for quantitative analysis of the proposed method in comparison with existing techniques. This chapter summarises contributions and discusses future work in the area of surface reconstruction in medical imaging. Section 6.1 provides a discussion of results and their implications. Current limitations and future work in this area are described in Section 6.2. Section 6.3 concludes the thesis.

6.1 Discussion of Results

The main contribution of this thesis is a proof of concept application of DTW to point correspondence in surface reconstruction in medical imaging. Shortest Euclidean distance is used as a cost function to find an alignment between points in contours, which is in turn used to generate a triangulation between contours. This thesis also proposes a rule-based contour correspondence method. The combination of these two methods allows for a full contour-based surface reconstruction technique. With

the addition of the proposed approach to merging contours, triangulations are successfully created for branching cases. The method was able to reproduce ground truth meshes that tested common issues in reconstruction. With greater contour correspondence methods, this method shows promise for use in the 3D reconstruction of complex anatomical structures from 2D slices. Point correspondence and contour correspondence have been implemented in a modular fashion, allowing for independent development and improvement.

Hausdorff distance and ground truth testing are used to provide a quantitative, comparative analysis between surface reconstruction methods. The proposed method shows lower average symmetric Hausdorff distance where there was less sampled model information. The comparative study presented in this thesis did not cover every surface reconstruction technique available. However, this work lays out foundations for further quantitative analysis in contour-based surface reconstruction. Ground truth testing which provides contour-like test data sets can be used to show reconstruction issues early in development, allowing for test-driven development in contour-based surface reconstruction.

The comparative analysis in Section 5.1 found that the proposed technique for surface reconstruction generally performed well with low amounts of input sample data in ground truth testing. The proposed method also reconstructed models with competitive distances to meshes given higher sampling rates in comparison to other techniques. The ability to produce quality reconstructions in varied sample rates is essential as the size of anatomical structures varies. This resilience is especially relevant when attempting to reconstruct internal structure such as bronchial trees. Another notable aspect of the ground truth testing was that Screened Poisson, Marching Cubes (APSS) and Marching Cubes (RIMLS) tended to outperform Voronoi Filtering and the proposed contour-based technique where there were large amounts of sample data available. These techniques show much higher average Hausdorff distance measurements in low sample rates. The requirement for vertex normal information in reconstruction also restricts the applications of these techniques.

Screened Poisson surface reconstruction was observed to be relatively robust in comparison to other point-cloud surface reconstruction techniques. With low contour sampling rates the Hausdorff distance was generally higher than the proposed

method, but otherwise, it fared well. From visual inspection of generated models, observations show that all produced models from this technique are reasonably similar to the source model, with no apparent issues. One major downside to this technique is that it requires point normal information. With improved normal estimation this technique will be more competitive in applications for contour-based reconstruction at lower sampling rates, or where there are more complex structures. Another consideration that is important depending on the application is that this technique (much like the other tested surface fitting techniques) generates a reconstructed surface that is not guaranteed to pass through all source points. With the implementation of accurate point normal estimation, particular applications of medical imaging may find use in approximations of the source data generated by Screened Poisson surface reconstruction.

Although the marching cubes reconstruction of both APSS and RIMLS fitting techniques had the lowest Hausdorff distance in some cases, these are generally not recommended as they did not handle lower contour sampling rates well. As well as having the highest Hausdorff distance in the majority of test meshes with low contour sampling rates, the meshes produced in these cases contained visible issues in reconstruction (Figure 5.3). The inconsistency of reconstruction accuracy for these techniques makes it generally unsuitable for applications in medical imaging. Voronoi Filtering also produced low average Hausdorff distances in some cases but showed relatively high average Hausdorff distance in more complex cases. This is attributed to the technique omitting features from the surface where data was undersampled. This inability to handle undersampled data reflects on previous observations in literature[24].

Section 5.2 demonstrates geometry information for reconstructions with various techniques. The proposed technique showed a smaller number of faces for varying contour sampling rates in comparison to other techniques. Despite showing the lowest average Hausdorff distance with only ten samples, this technique was able to represent an original test model with only 1894 faces. For the same test configuration, Screened Poisson showed the next lowest average symmetric Hausdorff distance. In comparison, this generated 5093 faces for the same model.

Contour correspondence of the proposed method was also evaluated in Section 5.3. This evaluation demonstrates the successes of the proposed method in the reconstruction of simpler branching structures. However, issues were noted where contours were considerably distant from each other between slices, and with the matching of too many contours where structures remained close after bifurcation takes place. The rule-based contour correspondence works well as a proof of concept, but the accuracy of contour correspondence in edge cases requires further improvement. Reconstruction of real data using the proposed surface reconstruction technique is demonstrated in Section 5.6, showing a logical triangulation for a branching case.

Brief running-time performance measurements are given in Section 5.4. As was expected, the largest contribution to implementation running time of the proposed method for large contours was generally the point correspondence stage of reconstruction. The input size of contours directly influences this stage of reconstruction. Generally, the time taken for contours below the size of 100 is negligible. However, when contour sizes start to near 1000 points, there is a considerable amount of time taken. Contour correspondence performance trends demonstrated how running time is affected by the number of contours on each slice.

In general, two broad approaches to this problem are considered: contour-based and point cloud surface reconstruction. By using contour information and making assumptions based on the desired application, contour-based approaches (such as the proposed method) provide a more accurate solution where there is little information about the source mesh. However, these approaches introduce a plethora of edge cases, leading to complex solutions. The proposed method does some edge case handling, but issues with correspondence and triangulation remain. Point-cloud surface reconstruction techniques provide a generalised solution to the reconstruction of unordered point clouds. The approach requires few prior assumptions about source data and does not require the handling of edge cases but does not provide sufficient accuracy with smaller samples.

6.2 Future Work

Currently, the proposed solution uses a naive implementation of DTW which uses no global constraints and a cost function that relies on Euclidean distance. Choice of boundary conditions, cost function, and global constraints are all variable. The selection of boundary conditions and cost function were a result of early preliminary testing. Further investigation into an ideal choice of the cost function and global constraint may provide further improvements to accuracy and the overall quality of reconstruction in general. An in-depth evaluation of the use of global constraints in DTW for this application should be done.

Additionally, the contour-merging implementation is only in a proof-of-concept stage. While this technique allows for correct removal of self-intersection triangles and has been shown to work with specific cases, some contour configurations are not handled. The handling of edge-case configurations with contour merging is left as future work.

Secondly, the proposed rule-based contour correspondence method has considerable limitations. This contour correspondence method works well enough to extensively test the point correspondence phase of reconstruction. The observed issues with this technique encourage further investigation into additional constraints or alternative methods for contour correspondence. A potential approach to investigate would be the representation of contour correspondence as a tree-like structure as it will allow for more content-aware ambiguity resolution. The optimal configuration of DTW for this particular application, coupled with more accurate contour correspondence techniques would allow for higher quality surface reconstructions.

With a more accurate implementation of the proposed reconstruction method, efficiency would need further evaluation and improved accordingly. Since the naive implementation of DTW was observed to be the most significant contributor to running time, more efficient implementations of this may be explored. When more efficient implementations of the proposed method are established, parallelisation may be used to further improve running-time performance. Implementation of a fully automatic surface reconstruction algorithm in a parallel manner without loss of accuracy would further advance indirect volume rendering research.

Lastly, the test generation method used can be further improved and automated. Test generation was performed with Blender3D to create models which represent specific scenarios encountered in surface reconstruction. Previous work has been done on tools for generation of branching structures by Pluta, Janaszewski, and Postolski [19]. Combination of a method similar to this with a test contour generation method such as the one proposed may allow for the generation of high-quality test data. An automatic tool such as this may prove useful for quick early analysis without access to any real data. Currently, the technique is limited to test data set creation from piecewise planar meshes. Extension of the proposed contour generation framework to support the use of implicit surfaces as input models may also prove useful.

6.3 Conclusion

With the introduction of modern applications for indirect volume rendering in medical imaging, previous research in contour-based reconstruction gains further use. Previously, literature has acknowledged a lack of both performance and accuracy of contour-based reconstruction techniques. For the specific application of medical imaging, there is lack of in-depth comparative analysis for surface reconstruction, despite the plethora of techniques available. The high quality of in-depth analyses in surface reconstruction for other applications creates a desire for such a comparative analysis for applications in medical imaging.

The introduction of DTW for use in point alignment for contour-based surface reconstruction is a step towards greater accuracy in indirect volume rendering. With the other minor improvements introduced in the reconstruction method proposed in this thesis, the implementation of DTW can find point alignments between contours that are logically consistent with real anatomical structures. The use of both ground-truth testing and reconstruction of real data demonstrates this logical reconstruction of contours.

This research also makes steps towards providing greater comparative analysis

in surface reconstruction for medical imaging. Ground truth based testing specifically for use with contour-based reconstruction is made possible with the introduction of the contour generation method proposed in this thesis. With this ground truth testing technique, test models that are directly applicable to scenarios in medical imaging are reconstructed and compared. The collected data showed that proposed contour-based surface reconstruction technique show resilience to the effects of low sample rates in source data when compared to free and open source implementations of point-cloud surface reconstruction algorithms. Further comparative analysis remains as future work, as there is a vast amount of reconstruction techniques available for comparison.

This thesis investigated DTW for applications in contour-based surface reconstruction for medical imaging, as well as comparative analyses for applications in this area. A full contour-based surface reconstruction method was created using DTW with a demonstrable ability to handle branching contours. Further venues to improving accuracy in the proposed technique are covered extensively and are left as future work. Running time performance of the proposed method was also measured. These measurements found that point correspondence running time is influenced by the size of input contours, and that contour correspondence running time is influenced by the number of contours on each slice. Further efficiency improvements to these stages and parallelisation is also left as future work.

Bibliography

[1] J. Meyer-Spradow, T. Ropinski, J. Mensmann, and K. Hinrichs, "Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations", *IEEE Computer Graphics and Applications*, vol. 29, no. 6, pp. 6–13, 2009.

[2] E. Sundén, P. Steneteg, S. Kottravel, D. Jonsson, R. Englund, M. Falk, and T. Ropinski, "Inviwo-an extensible, multi-purpose visualization framework", in *Scientific Visualization Conference (SciVis), 2015 IEEE*, IEEE, 2015, pp. 163–164.

[3] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: An open-source mesh processing tool.", in *Eurographics Italian Chapter Conference*, vol. 2008, 2008, pp. 129–136.

[4] B. Preim and C. P. Botha, *Visual computing for medicine: Theory, algorithms, and applications*. Newnes, 2013.

[5] M Balsa Rodríguez, E. Gobbetti, J. Iglesias Guitián, M. Makhinya, F. Marton, R. Pajarola, and S. K. Suter, "State-of-the-art in compressed gpu-based direct volume rendering", in *Computer Graphics Forum*, Wiley Online Library, vol. 33, 2014, pp. 77–100.

[6] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm", in *ACM siggraph computer graphics*, ACM, vol. 21, 1987, pp. 163–169.

[7] G. M. Nielson and B. Hamann, "The asymptotic decider: Resolving the ambiguity in marching cubes", in *Proceedings of the 2nd conference on Visualization'91*, IEEE Computer Society Press, 1991, pp. 83–91.

[8] R. Mukundan, "Contour based high resolution 3d mesh construction using hrct and mri stacks", *International Journal of Multimedia Data Engineering and Management (IJMDEM)*, vol. 8, no. 4, pp. 60–73, 2017.

[9] W. Birkfellner, *Applied medical image processing: A basic course*. CRC Press, 2016.

[10] G. Barequet and A. Vaxman, "Nonlinear interpolation between slices", *International Journal of Shape Modeling*, vol. 14, no. 01, pp. 39–60, 2008.

[11] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva, "A survey of surface reconstruction from point clouds", in *Computer Graphics Forum*, Wiley Online Library, vol. 36, 2017, pp. 301–329.

[12] P. Suetens, *Fundamentals of medical imaging*, ser. Cambridge medicine. Cambridge University Press, 2009, ISBN: 9780521519151.

[13] N. Sharma and L. M. Aggarwal, "Automated medical image segmentation techniques", *Journal of medical physics/Association of Medical Physicists of India*, vol. 35, no. 1, p. 3, 2010.

[14] J. C. Ross, R. S. J. Estépar, A. Díaz, C.-F. Westin, R. Kikinis, E. K. Silverman, and G. R. Washko, "Lung extraction, lobe segmentation and hierarchical region assessment for quantitative analysis on high resolution computed tomography images", in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2009, pp. 690–698.

[15] J. Pu, J. Roos, A. Y. Chin, S. Napel, G. D. Rubin, and D. S. Paik, "Adaptive border marching algorithm: Automatic lung segmentation on chest ct images", *Computerized Medical Imaging and Graphics*, vol. 32, no. 6, pp. 452–462, 2008.

[16] M. Kanzaki, T. Kikkawa, K. Sakamoto, H. Maeda, N. Wachi, H. Komine, K. Oyama, M. Murasugi, and T. Onuki, "Three-dimensional simulation, surgical navigation and thoracoscopic lung resection", *Journal of surgical case reports*, vol. 2013, no. 3, rjt015, 2013.

[17] A. Fedorov, R. Beichel, J. Kalpathy-Cramer, J. Finet, J.-C. Fillion-Robin, S. Pujol, C. Bauer, D. Jennings, F. Fennessy, M. Sonka, *et al.*, "3d slicer as an image computing platform for the quantitative imaging network", *Magnetic resonance imaging*, vol. 30, no. 9, pp. 1323–1341, 2012.

[18] MeVisLab, *MeVisLab: Medical Image Processing and Visualization*, https://www.mevislab.de/, Last accessed 2019-01-07, 2019.

[19] K. Pluta, M. Janaszewski, and M. Postolski, "New algorithm for modeling of bronchial trees", *Image Processing & Communications*, vol. 17, no. 4, pp. 179–190, 2012.

[20] C. Montani, R. Scateni, and R. Scopigno, "A modified look-up table for implicit disambiguation of marching cubes", *The visual computer*, vol. 10, no. 6, pp. 353–355, 1994.

[21] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm", *Computers & Graphics*, vol. 30, no. 5, pp. 854–879, 2006.

[22] G. Guennebaud and M. Gross, "Algebraic point set surfaces", in *ACM Transactions on Graphics (TOG)*, ACM, vol. 26, 2007, p. 23.

[23] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression", in *Computer Graphics Forum*, Wiley Online Library, vol. 28, 2009, pp. 493–501.

[24] N. Amenta and M. Bern, "Surface reconstruction by voronoi filtering", *Discrete & Computational Geometry*, vol. 22, no. 4, pp. 481–504, 1999.

[25] N. Amenta, M. Bern, and D. Eppstein, "The crust and the β-skeleton: Combinatorial curve reconstruction", *Graphical models and image processing*, vol. 60, no. 2, pp. 125–135, 1998.

[26] G. Barequet, D. Shapiro, and A. Tal, "Multilevel sensitive reconstruction of polyhedral surfaces from parallel slices", *The Visual Computer*, vol. 16, no. 2, pp. 116–133, 2000.

[27] G. Barequet, M. T. Goodrich, A. Levi-Steiner, and D. Steiner, "Contour interpolation by straight skeletons", *Graphical Models*, vol. 66, no. 4, pp. 245–260, 2004.

[28] Y. Li, S. Belkasim, Y. Pan, D. Edwards, and B. Antonsen, "3d reconstruction using image contour data structure", in *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, IEEE, 2006, pp. 3292–3295.

[29] C. Suppan, "GPU Accelerated Reconstruction of 3D Meshes from HRCT Contours", Computer Science Department, University of Canterbury, Tech. Rep., 2017.

[30] C. I. Fetita and F. J. Preteux, "Three-dimensional reconstruction of human bronchial tree in hrct", in *Nonlinear Image Processing X*, International Society for Optics and Photonics, vol. 3646, 1999, pp. 281–295.

[31] B.-S. Shin and H. S. Jung, "Contour-based terrain model reconstruction using distance information", in *International Conference on Computational Science and Its Applications*, Springer, 2005, pp. 1177–1186.

[32] E. Kienel, M. Vančo, G. Brunnett, T. Kowalski, R. Clauß, and W. Knabe, "A framework for the visualization of cross sectional data in biomedical research", in *Visualization in Medicine and Life Sciences*, Springer, 2008, pp. 77–97.

[33] K. Sopabutra, P. Horkaewa, and D. Gansawat, "Interactive three–dimensional model reconstruction of ultrasonography", in *The Proceedings of the 3rd International Conference on Industrial Application Engineering 2015*, vol. 28, 2015.

[34] S. Miyawaki, M. H. Tawhai, E. A. Hoffman, S. E. Wenzel, and C.-L. Lin, "Automatic construction of subject-specific human airway geometry including trifurcations based on a ct-segmented airway skeleton and surface", *Biomechanics and modeling in mechanobiology*, vol. 16, no. 2, pp. 583–596, 2017.

[35] S. Akkouche and E. Galin, "Implicit surface reconstruction from contours", *The Visual Computer*, vol. 20, no. 6, pp. 392–401, 2004.

[36] J. Manson, G. Petrova, and S. Schaefer, "Streaming surface reconstruction using wavelets", in *Computer Graphics Forum*, Wiley Online Library, vol. 27, 2008, pp. 1411–1420.

[37] S. Moriconi, E. Scalco, S. Broggi, B Avuzzi, R Valdagni, and G. Rizzo, "High quality surface reconstruction in radiotherapy: Cross-sectional contours to 3d mesh using wavelets", in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, IEEE, 2015, pp. 4222–4225.

[38] I. Braude, J. Marker, K. Museth, J. Nissanov, and D. Breen, "Contour-based surface reconstruction using mpu implicit models", *Graphical models*, vol. 69, no. 2, pp. 139–157, 2007.

[39] G. Taubin, "Smooth signed distance surface reconstruction and applications", in *Iberoamerican Congress on Pattern Recognition*, Springer, 2012, pp. 38–45.

[40] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction", *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, p. 29, 2013.

[41] J. Digne, D. Cohen-Steiner, P. Alliez, F. De Goes, and M. Desbrun, "Feature-preserving surface reconstruction and simplification from defect-laden point sets", *Journal of mathematical imaging and vision*, vol. 48, no. 2, pp. 369–382, 2014.

[42] S. Xiong, J. Zhang, J. Zheng, J. Cai, and L. Liu, "Robust surface reconstruction via dictionary learning", *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, p. 201, 2014.

[43] N. J. Mitra, A. Nguyen, and L. Guibas, "Estimating surface normals in noisy point cloud data", *International Journal of Computational Geometry & Applications*, vol. 14, no. 04n05, pp. 261–276, 2004.

[44] K. Demarsin, D. Vanderstraeten, T. Volodine, and D. Roose, "Detection of closed sharp edges in point clouds using normal estimation and graph theory", *Computer-Aided Design*, vol. 39, no. 4, pp. 276–283, 2007.

[45] B. Li, R. Schnabel, R. Klein, Z. Cheng, G. Dang, and S. Jin, "Robust normal estimation for point clouds with sharp features", *Computers & Graphics*, vol. 34, no. 2, pp. 94–106, 2010.

[46] W.-C. Lin, C.-C. Liang, and C.-T. Chen, "Dynamic elastic interpolation for 3d medical image reconstruction from serial cross sections", *IEEE transactions on medical imaging*, vol. 7, no. 3, pp. 225–232, 1988.

[47] W. Barrett, E. Mortensen, and D. Taylor, "An image space algorithm for morphological contour interpolation", in *Graphics Interface*, CANADIAN INFORMATION PROCESSING SOCIETY, 1994, pp. 16–16.

[48] J.-F. Guo, Y.-L. Cai, and Y.-P. Wang, "Morphology-based interpolation for 3d medical image reconstruction", *Computerized Medical Imaging and Graphics*, vol. 19, no. 3, pp. 267–279, 1995.

[49] D. H. Frakes, C. P. Conrad, T. M. Healy, J. W. Monaco, M. Fogel, S. Sharma, M. J. Smith, and A. P. Yoganathan, "Application of an adaptive control grid interpolation technique to morphological vascular reconstruction", *IEEE Transactions on biomedical engineering*, vol. 50, no. 2, pp. 197–206, 2003.

[50] D. P. Mukherjee and N. Ray, "Contour interpolation using level-set analysis", *International Journal of Image and Graphics*, vol. 12, no. 01, p. 1 250 004, 2012.

[51] C. A. Ratanamahatana and E. Keogh, "Everything you know about dynamic time warping is wrong", in *Third workshop on mining temporal and sequential data*, Citeseer, 2004.

[52] D. F. Silva and G. E. Batista, "Speeding up all-pairwise dynamic time warping matrix calculation", in *Proceedings of the 2016 SIAM International Conference on Data Mining*, SIAM, 2016, pp. 837–845.

[53] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition", *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[54] F. Itakura, "Minimum prediction residual principle applied to speech recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 67–72, 1975.

[55] T. Prätzlich, J. Driedger, and M. Müller, "Memory-restricted multiscale dynamic time warping", in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, IEEE, 2016, pp. 569–573.

[56] T. M. Rath and R. Manmatha, "Word image matching using dynamic time warping", in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, IEEE, vol. 2, 2003, pp. II–II.

[57] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping", *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[58] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces", in *Computer Graphics Forum*, Wiley Online Library, vol. 17, 1998, pp. 167–174.

[59] S. Silva, J. Madeira, and B. S. Santos, "Polymeco-a polygonal mesh comparison tool", in *Information Visualisation, 2005. Proceedings. Ninth International Conference on*, IEEE, 2005, pp. 842–847.

[60] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "Mesh: Measuring errors between surfaces using the hausdorff distance", in *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, IEEE, vol. 1, 2002, pp. 705–708.

[61] A. Bellmann, O. Hellwich, V. Rodehorst, and U. Yilmaz, "A benchmarking dataset for performance evaluation of automatic surface reconstruction algorithms", in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, IEEE, 2007, pp. 1–8.

[62] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva, "A benchmark for surface reconstruction", *ACM Transactions on Graphics (TOG)*, vol. 32, no. 2, p. 20, 2013.

[63] T. Wiemann, H. Annuth, K. Lingemann, and J. Hertzberg, "An extended evaluation of open source surface reconstruction software for robotic applications", *Journal of Intelligent & Robotic Systems*, vol. 77, no. 1, pp. 149–170, 2015.

[64] A. Thakur, A. G. Banerjee, and S. K. Gupta, "A survey of cad model simplification techniques for physics-based simulation applications", *Computer-Aided Design*, vol. 41, no. 2, pp. 65–80, 2009.

[65] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the hausdorff distance", *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[66] M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes", *IEEE Transaction on Visualization and Computer Graphics*, vol. 18, no. 6, pp. 914–924, 2012, http://doi.ieeecomputersociety.org/10.1109/T [Online]. Available: http://vcg.isti.cnr.it/Publications/2012/CCS12.

[67] J. Bowers, R. Wang, L.-Y. Wei, and D. Maletz, "Parallel poisson disk sampling with spectrum analysis on surfaces", *ACM Transactions on Graphics (TOG)*, vol. 29, no. 6, p. 166, 2010.

[68] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel, "Multi-level partition of unity implicits", in *ACM Transactions on Graphics (TOG)*, ACM, vol. 22, 2003, pp. 463–470.

[69] J. Lluch, R. Vivó, and C. Monserrat, "Modelling tree structures using a single polygonal mesh", *Graphical Models*, vol. 66, no. 2, pp. 89–101, 2004.

[70] H. Kitaoka, R. Takaki, and B. Suki, "A three-dimensional model of the human airway tree", *Journal of Applied Physiology*, vol. 87, no. 6, pp. 2207–2217, 1999.

[71] W. A. Mattingly, J. H. Chariker, R. Paris, D.-j. Chang, and J. R. Pani, "3d modeling of branching structures for anatomical instruction", *Journal of Visual Languages & Computing*, vol. 29, pp. 54–62, 2015.

[72] A Bourd, "The opencl specification, version: 2.2, document revision: 06", *Khronos OpenCL Working Group*, 2016.

[73] D. R. Kaeli, P. Mistry, D. Schaa, and D. P. Zhang, *Heterogeneous computing with opencl 2.0*. Morgan Kaufmann, 2015.

[74] Nvidia, *CUDA*, `https://developer.nvidia.com/cuda-zone/`, Last accessed 2018-07-07, 2018.

[75] S. Cook, *Cuda programming: A developer's guide to parallel computing with gpus*. Newnes, 2012.

[76] S. Mittal and J. S. Vetter, "A survey of cpu-gpu heterogeneous computing techniques", *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, p. 69, 2015.

[77] K. Karimi, N. G. Dickson, and F. Hamze, "A performance comparison of cuda and opencl", *ArXiv preprint arXiv:1005.2581*, 2010.

[78] J. Fang, A. L. Varbanescu, and H. Sips, "A comprehensive performance comparison of cuda and opencl", in *Parallel Processing (ICPP), 2011 International Conference on*, IEEE, 2011, pp. 216–225.

[79] S. Memeti, L. Li, S. Pllana, J. Kołodziej, and C. Kessler, "Benchmarking opencl, openacc, openmp, and cuda: Programming productivity, performance, and energy consumption", in *Proceedings of the 2017 Workshop on Adaptive Resource Management and Scheduling for Cloud Computing*, ACM, 2017, pp. 1–6.

[80] M. A. Alsmirat, Y. Jararweh, M. Al-Ayyoub, M. A. Shehab, and B. B. Gupta, "Accelerating compute intensive medical imaging segmentation algorithms using hybrid cpu-gpu implementations", *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 3537–3555, 2017.

[81] J. I. Agulleiro, F. Vazquez, E. M. Garzon, and J. J. Fernandez, "Hybrid computing: Cpu+ gpu co-processing and its application to tomographic reconstruction", *Ultramicroscopy*, vol. 115, pp. 109–114, 2012.

[82] M. Bolitho, M. Kazhdan, R. Burns, and H. Hoppe, "Parallel poisson surface reconstruction", in *International symposium on visual computing*, Springer, 2009, pp. 678–689.

[83] B. Foundation, *Blender - Free and Open 3D Creation Software*, `https://www.blender.org/`, Last accessed 2019-02-05, 2019.

[84] T. Möller, "A fast triangle-triangle intersection test", *Journal of graphics tools*, vol. 2, no. 2, pp. 25–30, 1997.

[85] J. F. Hughes, A. Van Dam, J. D. Foley, M. McGuire, S. K. Feiner, D. F. Sklar, and K. Akeley, *Computer graphics: Principles and practice*. Pearson Education, 2014.

[86] G.-T. Creation, *OpenGL Mathematics*, `https://glm.g-truc.net/`, Last accessed 2019-02-08, 2018.

[87] Boost, *Boost C++ Libraries*, `http://www.boost.org/`, Last accessed 2018-11-09, 2018.